5-2016

Production and inventory control in complex production systems using approximate dynamic programming.

Han Wu

PRODUCTION AND INVENTORY CONTROL IN COMPLEX PRODUCTION SYSTEMS USING APPROXIMATE DYNAMIC PROGRAMMING

By

Han Wu

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Industrial Engineering

Department of Industrial Engineering
University of Louisville
Louisville, Kentucky

May 2016

PRODUCTION AND INVENTORY CONTROL IN COMPLEX PRODUCTION SYSTEMS USING APPROXIMATE DYNAMIC PROGRAMMING

By

Han Wu

A Dissertation Approved On

November 30, 2015

by the following Dissertation Committee:

_____

Dr. Gerald Evans Dissertation Director

_____

Dr. Kihwan Bae Dissertation Co-Director

_____

Dr. Suraj M. Alexander

_____

Dr. Lihui Bai

_____

Dr. Dar-jen Chang

ACKNOWLEDGEMENTS

It is with great pleasure that I take this opportunity, to thank the people who have made my dissertation possible.

Foremost, I would like to express my sincerest gratitude to my advisor Dr. Gerald W. Evans for his continuous support in my Phd study and research. Also for his understanding, patience, and enthusiasm. His guidance helped me throughout my research and writing of this dissertation.

Secondly, I would like to thank my dissertation co-advisor Dr. Ki-Hwan Gabriel Bae. His brilliant suggestions were indispensable for my Phd research.

I would also like to thank Dr. Suraj Alexander and Dr. Lihui Bai. I am very grateful for their suggestions to my research and help for the course work during the early stages of my Phd study.

Moreover, I wish to thank Dr. Dar-Jen Chang for his interest in my research and kindness in sharing his knowledge outsides the Industrial Engineering area to inspire my research.

Finally, I need to thank for my parents, my girlfriend Chen Zhou, and all my friends in the United States and China for their continuous support and encouragement.

I dedicate this dissertation to those who have helped me reach this point, and those who will help me in my future life.

ABSTRACT

PRODUCTION AND INVENTORY CONTROL IN COMPLEX PRODUCTION SYSTEMS USING APPROXIMATE DYNAMIC PROGRAMMING

Han Wu

November 30, 2015

Production systems focus not only on providing enough product to supply the market, but also on delivering the right product at the right price, while lowering the cost during the production process. The dynamics and uncertainties of modern production systems and the requirements of fast response often make its design and operation very complex. Thus, analytical models, such as those involving the use of dynamic programming, may fail to generate an optimal control policy for modern production systems.

Modern production systems are often in possession of the features that allow them to produce various types of product through multiple working stations interacting with each other. The production process is usually divided into several stages, thus a number of intermediate components (WIP) are made to stock and wait to be handled by the next production stage. In particular, development of an

efficient production and inventory control policy for such production systems is difficult, since the uncertain demand, system dynamics and large changeover times at the work stations cause significant problems. Also, due to the large state and action space, the controlling problems of modern production systems often suffer from the "curse of dimensionality".

In this dissertation, we generalize problem associated with the controlling of production systems as a stochastic-dynamic decision making problem for multiple machines with intermediate products, and compare it to the classic Stochastic Economic Lot Scheduling Problem.

To address the complexity optimizing control process of systems facing uncertain demands, system dynamics and large changeover time, we first proposed an adjusted (s, S) policy, and optimize it through the use of a simulation model. Also, two advanced Approximate Dynamic Programming (ADP) methods are proposed to handle the "curse of dimensionality", thus helping the system to make decisions at particular states. One of the ADP methods is based on a set of linear regression models to approximate the value function of the state. The other ADP method is based on an Artificial Neural Network model which is designed to capture the features of this problem and also embed the adjusted (s, S) policy.

The proposed methods are tested on a small numerical example and also applied to an assembly line for dishwashers which requires multiple types of wire racks that must be fabricated and coated at different work centers before supplying

the assembly lines. The near optimal production and inventory control policies are developed through the proposed methods. These proposed methods, especially the ADP methods, can be extended to any similar production system, or solve similar problems in other fields.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.A Background

The basic focus of most organizations is to provide goods and services or more generally to fulfill the needs of customers, and meanwhile lower the costs of these goods and services to enhance the competitive strength of the company. This is accomplished through the use of a production system, which can be defined as the set of resources and procedures involved in converting raw material into products and delivering them to customers (Askin and Goldberg, 2002). An enterprise which provides better products at lower cost than their competitors can make more profit than these competitors. One approach for reducing production cost involves improving the control of the production system.

Inventory cost always occupies a substantial portion of the manufacturing cost, often 20% or more (Askin and Goldberg, 2002). Hence, the development of an optimal control policy would be an overall consideration of the interactions between the elements in the system, rather than only considering the reduction of the energy cost, machine idle time or the inventory level independently.

Production systems are becoming increasingly complex, as a result of the

various types of systems (e.g., parallel, rework, and JIT structures) in existence and the dynamics of their operations (e.g., involving machine breakdowns and changeovers). These complexities make such systems extremely difficult to design and operate. Although researchers have attempted to formulate and analyze these complex production systems via analytical models, it is rather difficult to capture the dynamics and uncertainties with these analytic models and analyze the corresponding systems accurately.

Flexibility of machines or work stations is one of the important characteristics in modern manufacturing systems. Investing in a machine which is flexible to produce more than one type of product or for multiple processes usually will reduce cost. Moreover, the smaller batches of various product types not only meets the increasing need for current customers, but could also reduce the inventory cost considerably. However, the significant changeover times between different kinds of products would always make a optimal control scheme difficult to develop for the production system.

Many companies have changed their production control strategies from a "push" strategy (e.g., MRP, MRP-II, and ERP) to a "pull" (e.g., Kanban and CONWIP). Compared to the "push" strategies, the "pull" strategies maintain lower work-in-process (WIP) inventory levels in the system, thus requiring less space for accommodating fluctuation and minimizing congestion. Although the "pull" strategy can significantly reduce the cost waste on inventory, the uncertain demands of customers make the inventory control of modern manufacturing systems much

more complicated. The managers will need some refined policies to better control the production systems and to respond to customers' demands faster and more precisely.

The stochastic economic lot scheduling problem, (SELSP), is a dynamic production problem involving risk and uncertainty. The SELSP deals with the make-to-stock production of multiple standardized products on a single machine with limited capacity under random demands, possibly random setup times and possibly random production times (Winands et al., 2010). Even the deterministic economic lot scheduling problem has been proven to be NP-hard, and due to the complexity and lack of an analytical method, the research on the SELSP started relatively recently (No research found in the literature mentioned it before Winands et al. 2010). However, there are many production systems with multiple machines producing multiple products which have problems similar to the SELSP. Finding an optimal or near-optimal policy for these systems would be more difficult than solving the typical SELSP.

I.B Motivation

To develop a method to find optimal actions to control a dynamical system with numerous states and stochastic aspects is an interesting topic not only in the area of production, but also in the areas of finance, transportation, energy, medical, etc. We might call this kind of problem as a stochastic-dynamic decision problem. In stochastic-dynamic decision problems, we need to make a series of decisions over

multiple periods facing some level of uncertainty. Normally, these problems can be modeled as Markov Decision Processes (MDPs). However, when the space of states, actions and/or outcomes becomes large, the difficulty involved in solving these problems increases dramatically. This situation is known to be the *"three curses of dimensionality"*.

No doubt, the classic SELSP suffers from the three curses of dimensionality even for a small problem with only a few product types. (One needs to deal with the huge space of states which arises from the combination of machine status and inventory level of each product), not mention the fact that production system has multiple machines and product types.

To break down the notorious three curses of dimensionality, a method called approximate dynamic programming (ADP) has been studied by several scholars. ADP arises from computer science, especially the realm of artificial intelligence (Powell, 2011). Compared to the "backward" strategy of solving formal deterministic dynamic programming problems, ADP steps forward and uses iterative algorithms to estimate a value function, then finally solve the problem which is restricted by the random environment.

Approximate dynamic programming is emerging as a powerful tool for certain classes of multistage stochastic, dynamic problems that arise in operations research. It has been applied to a wide range of problems spanning complex financial management problems, dynamic routing and scheduling, machine scheduling, energy management, health resource management, and large-scale fleet management

problems. It offers a modeling framework that is extremely flexible, making it possible to combine the strengths of simulation with the intelligence of optimization (Powell, 2008). ADP does seem to be an attractive methodology for generating a good control policy for a complex production system.

I.C Dissertation Organization

The dissertation is organized as follows:

Chapeter II first describes the stochastic economic lot scheduling problem (SELSP). Then the multiple machines with intermediate products problem is introduced with an actual case from General Electric's dishwasher production system. The complexity of the problem is also discussed in chapter II. Chapter III reviews the literature of classic methods for solving SELSP or similar problems. The methodology of ADP is systematically introduced in chapter IV. A basic algorithm of ADP is presented in section IV.A.1. Several value function approximation schemes are presented in section IV.C and an simple example is shown in section IV.D. The notation and assumptions are introduced in chapter V, with the formulation of a general model for the problem discussed in section II.B. Three methods are proposed in chapter VI: section VI.A discusses a method to solve the problem by using optimization via simulation through an adjusted (s, S) policy; section VI.B develops an approximate dynamic programming method by using a set of linear regression models to approximate the value function of the system; section VI.C proposes an artificial neural network model to capture the feature of the

problem and also combines the adjusted (s, S) policy to evaluate the state value, thus controlling the production system. Two approximate dynamic programming algorithms are developed in section VI.B.4 and in section VI.C.4 to perform the learning process for the two approximation schemes discussed in section VI.B.1 and in section VI.C.2. An simple numerical example is set up and the methods proposed are studied in chapter VII. A real case arising from General Electric's dishwasher production system is solved in chapter VIII. Chapter IX discusses the contribution and conclusions of this research, along with future research directions.

CHAPTER II

PROBLEM STATEMENT

For most production systems, the complexity associated with the problem of developing the control and inventory policy is related to the dynamics and uncertainties in the systems. For instance, a machine or production line may need significant time to switch from producing one type of product to another, and the demand of the product may be uncertain; moreover, machines in the system might fail in a random fashion. Although the manufacturing environment may differ from one industry to another, one needs to deal with the problem of "when to produce how many/much of what" in order to guarantee a properly working production system. However, the dynamics and uncertainties associated with the system make a good answer to the problem very difficult to find. Many technologies have been developed to find a good control and inventory policies for production systems under different manufacturing environments. This research deals with the problem of how to control a production system in order to produce multiple products with significant machine setup times under a stochastic environment.

II.A Stochastic Economic Lot Scheduling Problem

II.A.1 General Description

The stochastic economic lot scheduling problem (SELSP) is the problem of scheduling production of multiple products, each with random demand, in a single facility that has limited production capacity and significant changeover times between products (Sox et al., 1999). Figure 1 shows a general configuration of a production system associated with the SELSP.

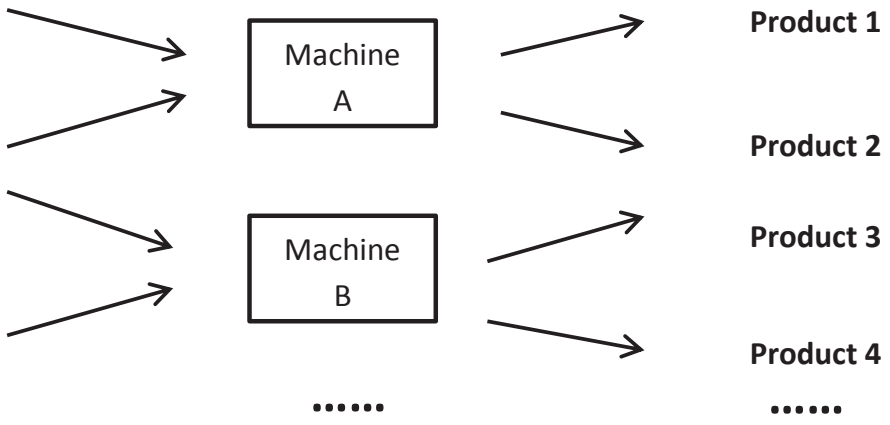

Figure 1. The General System of the Stochastic Lot Scheduling Problem.

The SELSP is one of the classic problems in production planning research. Applications include glass and paper production, injection molding, metal stamping, semi-continuous chemical processes, and bulk production of consumer products such as detergents and beers, etc. Any production process with significant changeover times between products benefits from an effective scheduling system.

II.A.2 Complexity of SELSP

The deterministic version of economic lot scheduling problem (ELSP) has received much attention in the literature in recent decades. The ELSP has been proven to be NP-hard (Hsu, 1983). Two types of approaches have been developed to solve the ELSP: an exact type of approach with optimal solutions for restricted problems and a heuristic approach with good solutions for the general problem (Winands et al., 2010). Both approaches derive a rigid cyclic schedule, which will be strictly followed until the end of the planning horizon (Gascon et al., 1994). However, Gallego (1990) has argued that the solution methods of the ELSP can only be applied in an ideal production environment, where machines are perfectly reliable, setup and production rates are constant, raw material and tools are always available, demand is known and initial inventories are provided. Due to the severe production environment in real manufacturing systems, the deterministic problem should be extended to a stochastic version, the SELSP.

Compared with the deterministic ELSP, the stochastic nature of demand added in SELSP makes this problem much more complex. In the SELSP, the production capacity is limited, and it need to be allocated among the products; however, the randomness of the demands means that the allocation of the capacity must be dynamic. The dynamic allocation of production capacity is dependent on the inventory levels of the various products in the system, since they share the same inventory location. The various types of products compete for production capacity,

thus a much higher safety stock level would be needed to maintain a specific service level above what would be required using a dedicated production facility for each product (Sox et al., 1999).

Winands et al. (2010) have pointed out that the presence of change-over times in combination with the stochastic environment are the key complicating factors of the SELSP. Due to this unavoidable change-over time, there is a significant delay when the system switches from producing one product to another. If the system continues to spend too much time on the production of one product to replenish its inventory level, the depletion of inventory for other products would leave the system in a potentially less favorable state. The drawback of the situation is: that to respond to the random demand in the system, one needs to shorten the cycle length for each product; thus frequent production opportunities for the various products. However, shortening the production cycle length would lead the system to perform changeovers too frequently, which would waste a lot time, or even make the system state blocked in change-over so that it can not fulfill the demand. Therefore, to develop a valid control policy for the SELSP, both production and inventory must be considered simultaneously and the decisions must be effective for a long time horizons.

II.A.3 Summary

The system associated with an SELSP can be summarized as follow Winands et al. (2010):

- A production system with a single machine which can produce multiple products, but only one at a time; the raw material is unlimited, while the stock space is limited.

- Demands for the various products arrive according to stationary and mutually independent stochastic processes. Demand that cannot be satisfied directly from stock is either lost or backlogged until the product becomes available after production.

- The individual products are produced in a make-to-stock fashion with possibly stochastic production times. A setup time (that is possibly stochastic as well) occurs before the start of the production of a product.

Due to the desire for efficient control of the production process, production and setup times are often (almost) deterministic. The setups are, furthermore, independent of the demand processes, production times and other setup times. The main objective of the SELSP is to minimize the total expected costs per unit of time over a planning horizon, which can either be finite or infinite. Besides the total costs, other quantities of interest could, for example, be the fraction of time that is lost due to setups, the fill rate (the fraction of demand satisfied directly from stock), the average stock level or the average waiting time of customers Winands et al. (2010). In the SELSP, a production policy should be able to decide whether to continue to produce the current product, whether to switch to another product or whether to idle the machine.

II.B Multiple Machines with Intermediate Products

II.B.1 GE Production System

General Electric's Appliance Park, located in Louisville, Kentucky, produces various appliances, including dishwashers. The dishwasher production system has three fabrication centers: 1) a center (denoted as FL) to produce three types of lower dishwasher racks (denoted as types A, B, and BXL), 2) a center (denoted as FU1) to produce four types of upper dishwasher racks (denoted as A1, B1, B2, B3), and 3) a center (denoted as FU2) to produce two additional types of upper racks (denoted as C2 and C4); the system also contains two coating centers: one for nylon coating (denoted as Nylon) which has three colors (Color A, Color B and Color C), and one for PVC coating (denoted as PVC) which only has one color (Color D). The five work centers (FL, FU1, FU2, Nylon and PVC) constitute the production system to produce the wire racks which supply dishwasher assembly lines. The facilities layout and production process are illustrated in Figure 2.

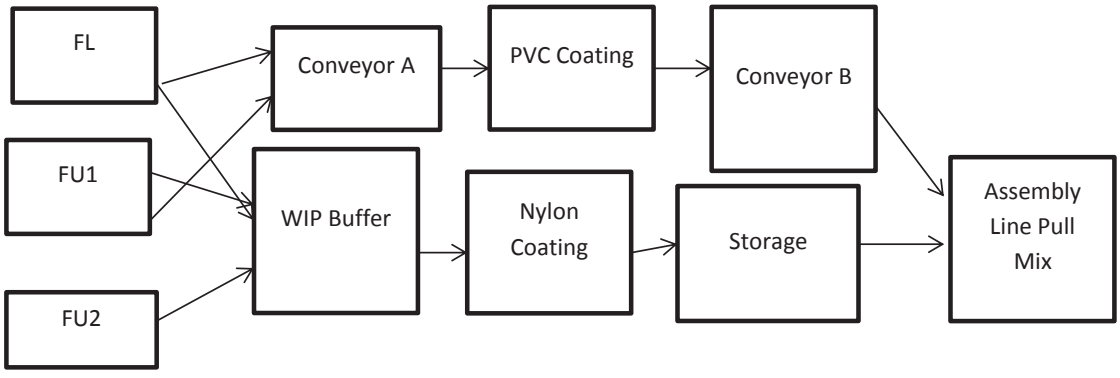

Figure 2. Production Process and Facilities Layout.

This production system has 9 types of WIP racks fabricated separately at the three fabrication centers (see Table 1). These WIP racks are stored in the WIP buffer area and conveyor A which can be treated as a buffer between the three fabrication centers and the two coating centers. Thirteen types of coated racks, which can be considered as the finished products, are stored in the storage area and conveyor B serves as a buffer in front of the assembly lines as well. For each type of coated rack, the arrival of the demand from the assembly lines corresponds to a Poisson process, and the demand size of each arrival is normal distribution.(see Table 2).

TABLE 1

Rack Model or Color at each Work Center.

Work Center	FL	FU1	FU2	PVC	Nylon
	A	A1	C2	Color D	Color A
Rack Model or	B	B1	C4		Color B
Color Types	BXL	B2			Color C
		B3			

TABLE 2

13 Types of Coated Rack.

Fab Center	Rack #	Model	Coating Type	Color
FL	1	A	Nylon	Color A
	2	B	PVC	Color D
	3	B	Nylon	Color A
	4	BXL	Nylon	Color B
	5	BXL	Nylon	Color C
FU1	6	A1	Nylon	Color A
	7	B1	PVC	Color D
	8	B2	Nylon	Color A
	9	B3	Nylon	Color B
	10	B3	Nylon	Color C
FU2	11	C2	Nylon	Color B
	12	C4	Nylon	Color B
	13	C4	Nylon	Color C

The different models of fabricated racks and the coating colors associated with the five work centers are summarized in Table 1. The estimated time intervals for making changeovers from one model or color to another at each work center were provided by GE.

The current dishwasher wire rack production policy in use at GE is a "push" strategy. The multiple types of racks are produced and a changeover is performed according to a production plan developed daily by the production manager. The fabricated and coated racks are stacked in the WIP buffer, storage areas and the two conveyors. GE wishes to develop a production control policy which can react to

and fill the assembly line demand, in order to minimize the number of changeovers and reduce the inventory levels significantly.

II.B.2 Generalization

Obviously, the problem described in section II.B.1 has many properties similar to the SELSP:

- It is a make-to-stock production system with multiple products.
- One machine (or work center) can produce different types of products, but have a significant setup time from product to product.
- Each machine (or work center) can only produce one type of product at a time.
- Demands for products are random.
- The stock space is limited.

However, there are two major differences between the SELSP and the GE problem:

- There are several machines (or work centers) associated with the GE problem, instead of single machine in the production system.
- The production process of the final product is two stages divided by the intermediate components.

The complexities of adding the two differences will be discussed in section II.B.3. In summary, the GE problem is a SELSP with multiple machines and intermediate components. The problem can be generalized by following:

Consider a two-stage production system, each stage have one or more parallel machines to produce different types of products. In the first stage, the products are intermediate components (also referred to as intermediate products) which are produced to stock for wait to be used by the second stage. The second stage will deplete the stock of intermediate components to produce the finished product to fulfill the uncertain demands from the customers. All the machines in both of the two stages can process more than one type of product, but only one type at a time. The switch from one product to another on a machine would cause to a significant change-over time. During this changeover time, the machine can not produce additional items. There are two limited inventory spaces in the production system, one for the intermediate components, and the other for finished product. The demands for the finished products arrive according to stationary and mutually independent stochastic processes, and the demand sizes are random. Demand that cannot be satisfied directly from stock of the finished products is either lost or backlogged until the product becomes available after production. Figure 3 gives a visual depiction of the production system in the problem.

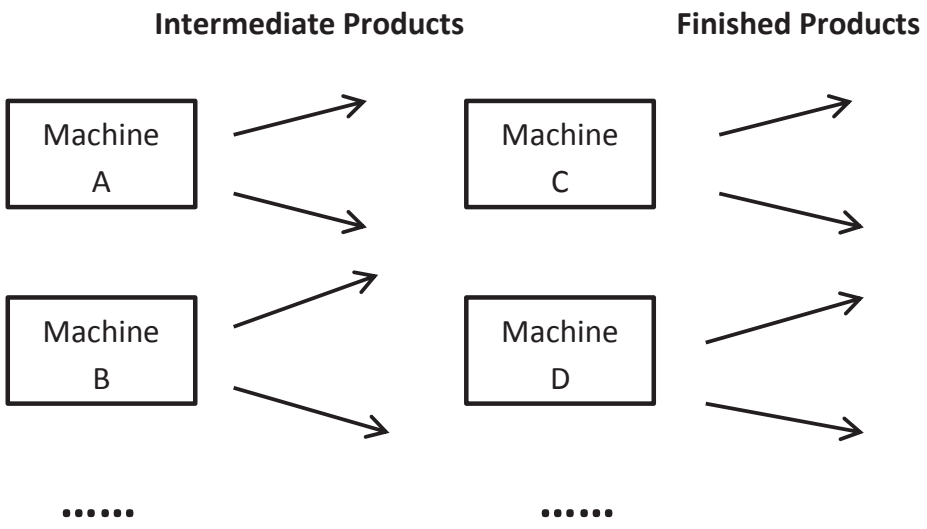

Figure 3. The Generalization of the GE System

The main objective of this problem is to generate a control policy to minimize the total expected costs per unit of time over a planning horizon. Moreover, one may also be interested in the inventory levels for both of the intermediate components and the finished products, the percentage of unmet demands for the finished products, the number of change-overs, etc.

The GE problem is very common in many industries. For example, a chemical company may have a system consisting of several reaction vessels to produce different types of product, but if they want to change the product type for one vessel, the time to rinse is unavoidable. An automobile plant may have several assembly lines to supply their painting workshop; the different models produced in these assembly lines combined with the color in the painting workshop results in a variety of unique products; however, the changeover and preparation time for such industry is significant. Consequently, there would be much potential benefit if a

good general solution method could be developed for this kind of problem.

II.B.3 Complexity

Compared with classic SELSP, the GE problem is much more complex, since it has more than one machine and intermediate components. To generate the control policy for the production system, one needs to consider the number of possible states in which the system could exist. The production status of the system is described by the combination of discrete machine states. If one more machine is added in the system, the number of the production states will definitely increase exponentially. In the GE problem, besides the inventory pool for the finished product, another inventory pool is required for the intermediate components. How to allocate the capacity among the intermediate components is a new challenge for this problem. The intermediate components divide the production process into two stages, and they are not only the finished product for the first stage but also the required materials for the next stages. Therefore, the inventory levels of the intermediate components can not become negative and the unsatisfied demands of intermediate components can neither be considered as lost sale nor back-order; moreover, the lack of intermediate components would affect the production for the next stage which is producing the finished products to fulfill the customer demands. The existence of intermediate components not only doubles the effort needed for allocating the production capacity, but also increases the overall dynamic complexity of the production system.

To generate a good control policy for the GE problem, besides dealing with the balance of shorter production cycles and the frequencies of change-overs, one also needs to consider the interactions of the machines in the systems, as well as the dynamics of the inventory levels for the intermediate components. Thus, the production control policy needs to be able to control every single machine on an overall view to the system.

CHAPTER III

LITERATURE REVIEW

Chapter III will focus on surveying the classic literature related to this research. We will start with Stochastic Economic Lot Scheduling Problem.

The earliest survey paper on the SELSP appeared in Sox et al. (1999). In this paper, Sox et al. classified the literature of the SELSP based on the modeling methods introduced by academicians. Following the publication of the paper by Sox et al. (1999), a large number of papers related to the SELSP appeared in the open literature. For example, another survey paper was published by Winands et al. (2010), which introduced the new literature published during 1999-2010 and reclassified this literature based on the critical elements of production planning as seen by practitioners. More specifically, Winands et al. (2010) addressed the literature associated with the solution methods for the problems similar to SELSP but even more complex. From historical literature, those methods can be classified by the three different strategies for developing the control policy: sequence-cycle, base-stock and combination of sequence-cycle and base-stock.

III.A Sequence-cycle method

The sequence associated with a production control policy can be determined in a totally dynamic fashion.

Karmarkar and Yoo (1994) studied this kind of problems under the assumption of deterministic production and setup times with unmet demands considered to be "backlogged". They formulate the problem using a discrete-time stochastic dynamic programming model over a finite horizon under the assumption of a time-varying stochastic demand. In their paper, the problem was solved by Lagrangian relaxations with lower and upper bounds are provided for the original problem.

Qiu and Loulou (1995) modeled the SELSP as a continuous-time semi-Markov decision problem with an infinite time horizon. The current status of the machine and the inventory levels of the individual products constituted the state space. The demand for the products was assumed to be distributed according to a Poisson process. Qiu and Loulou solved their problem by using the successive approximations technique, in which the near-optimal policy can be extended and derived on a truncated finite state space.

Sox and Muckstadt (1997) modeled the SELSP as a finite-horizon discrete-time stochastic optimization problem under the assumption of overtime being available. A Lagrangian decomposition algorithm was developed to find an optimal or near-optimal solution for their problem.

The methods applied in the three papers of Karmarkar and Yoo (1994), Qiu and Loulou (1995), and Sox and Muckstadt (1997) were proven to be not very suitable for large-scale problems or more complex systems. Due to the curse of dimensionality, the solution procedure developed by Qiu and Loulou (1995) was found to be inefficient and inaccurate for large problems. Sox and Muckstadt (1997) assumed that a setup for a product is incurred even if the same product was produced in the preceding period. Extensive computational time would be required if the assumption were relaxed.

There is much more literature addressing this kind of problem by using the strategy of fixed-sequence rather than by using a totally dynamic one.

As early as 1988-1991, a dynamic cycle lengths heuristic was developed by Leachman et al. through (Leachman and Gascon, 1988) and (Leachman et al., 1991). They solved the uncertainties in the problem via a deterministic approach with the use of estimates. At first, they used the moving averages of the demand forecast to calculate the target cycle lengths in each review period. Then, the operational cycle lengths were determined by proportionally reducing the production quantities of all products in a cycle, while maintaining the fixed production sequence. Finally, when all products have sufficient stock, the idle periods are inserted in cycle if possible.

Fransoo (1992) studied a model similar to those in Leachman and Gascon (1988) and Leachman et al. (1991), and found that if one expects a product to be consumed as much as possible in the following cycle, the dynamic cycle lengths

heuristic would reduce the cycle length for this product in the current cycle, which would lead to an increase in the relative setup frequency and a decrease in the capacity available for production; thus, the future demand would be even more difficult to fulfill. Also, to improve on the heuristic proposed by Leachman et al. (1991), Fransoo developed an alternative heuristic which results in the cycle lengths becoming more stable.

Erkip et al. (2000) proposed a fixed cycle strategy, which fixed not only the sequence and the total cycle length, but also the available capacity for each individual product. Their strategy is modeled as a quasi−birth−death process, which can be solved numerically by a matrix-analytic method.

From 2000 to 2001, Markowitz et al. developed a solution method for SELSP motivated by the well-known heavy-traffic limit theorems in (Markowitz et al., 2000) and (Markowitz and Wein, 2001). In their model, a time-scale decomposition is produced, and the SELSP can be approximated by a diffusion control problem. Other recent work based on heavy-traffic analysis is by Bruin (2007), who presents a generating function approach for the fixed cycle strategy under general traffic settings.

III.B Base-stock method

The base-stock production strategies for these kinds of problems are normally developed through standard single product inventory control strategies such as (s,Q) or (s,S) policies.

Zipkin (1986) developed an (s, Q) policy to solve a model with Poisson demand processes and generally distributed setup and production times. Zipkin made the assumption that the production time of a batch is (nearly) independent of the size of this batch in this paper. In the developed policies, the batches for the various products are produced in a first come first served (FCFS) order. In a recent work, Winands et al. (2009) extended Zipkin's model (Zipkin, 1986) to consider a general (renewal) arrival process for demand, multiple parallel lines and various service measures. Also, the production times did not have to be independent of the batch sizes.

Beginning in 1994, Altiok and Shiue published a series of papers (see Altiok and Shiue, 1994, 1995 and 2000a) that implement an (s, S) policy to solve these kinds of problems. For example, Altiok and Shiue (1994) developed two priority rules to determine the sequence of product production when more than one of the product's inventory levels goes below their respective reorder points. The first one is the general priority rule: the machine will start to produce the highest priority product with inventory position below its reorder point, when the inventory position of the product currently set up reaches its base-stock level. The second priority rule is a cyclical one, since it goes into effect when the products' inventory levels go below their reorder points in a cyclical manner. In 1994, Altiok and Shiue analyzed the case of three products, and extended their model to the N products in Altiok and Shiue (2000a). In 1995, Altiok and Shiue analyzed a lost sales case under the additional assumptions of phase-type distributed production times and

exponentially distributed setup times.

Paternina-Arboleda and Das (2005) applied reinforcement learning with optimization via simulation via an approach to analyze a base stock policy: if the product currently being produced reaches its base-stock level, the machine is allowed to switch to produce another product or to remain idle until the next demand arrival period when any new action might need to be performed. However, the major drawback of Paternina-Arboleda and Das's approach is that the developed policy is very difficult to implement, since some data mining classification techniques need to be applied when looking for a (near) optimal policy.

Brander et al. developed an approximate method to determine the safety stocks and base-stock levels under given fixed production sequences (see Brander et al. 2005, and Brander and Forsberg 2006). To determine whether to idle the machine or to produce the next item in the sequence, they developed a control model. Their work is implemented through the use of simulation results to estimate the lot-size in a stochastic environment. It is also very interesting to mention that their works (Brander et al., 2005) showed that the lot-sizes determination decisions is of less importance than the sequencing decision.

III.C Combination of Sequence-cycle and Base-stock

One may want to utilize the advantages associated with both of the strategies of sequence-cycle and base-stock. Therefore, some literature which combines these strategies is reviewed in this section.

Frequently, the policy would be divided into two levels. Bourland and Yano (1994) developed a two-level hierarchical policy for the SELSP. Their strategy assumes that for each individual product a reorder point is given. In the policy, the upper level sub-policy determines the cyclic schedule, cycle length, stock levels and idle time by ignoring the uncertainty associated with the demand. The lower level is a control level that defines the control rule to follow the targets set by the upper level. In another paper, Bourland (1994) let the production quantity be determined by a match-up lot-sizing policy; such a match-up policy schedules production of a product so that the stock level at the planned completion time - and not necessarily at the actual completion time - of the production run is equal to the base-stock level. Bourland (1994) also mentioned that such a match-up policy follows the target cycles more effectively as compared to a standard base-stock policy.

Wagner and Smits also proposed a two-level policy to solve such kind of problems in 2004. The upper level for the policy will decide the optimal fixed cycle schedule by considering the expected setup and holding costs. The lower level will derive a periodic (R,S) policy, where the optimal base-stock levels are obtained by an algorithm developed by Smits et al. (2004).

Federgruen and Katalan computed the optimal base-stock levels by solving standard newsboy problems and constructing the optimal production sequence by approximation (see Federgruen and Katalan 1996b, Federgruen and Katalan 1996a, and Federgruen and Katalan 1998). In their research, they showed that total average costs only depend on the total idle time inserted in a cycle and not on the

complete vector of idle times.

Grasman et al. (2008) extend Federgruen and Katalan's model (Federgruen and Katalan, 1996b) by adding random yields for the cases of backlogging and lost sales. To obtain the optimal base-stock levels, they use the similar newsboy equations in case of backlogging, and a heuristic for approximation in case of lost sales.

Vaughan (2003) considered correlated demand, and used a base-stock strategy and a target cycle length to develop the policy. The policy will allow the machine to be idle when a cycle is ended within the target length, or else, the machine will start next cycle immediately. In this research, Vaughan point out that demand correlation will increase the variance of the cycle length and also cause the correlation between demand per period and the cycle length. This will lead to a higher variance of the total demand during a cycle and require larger safety stock levels.

As compared to the two-level policy, Gallego (1990) proposed a three-level production control policy. The production sequence, the production quantities and the idle times are constructed at the first level by a deterministic approach. The uncertainties in the problem are handled at the second level by a policy which can recover the target schedule at minimal excess over average costs after a random event happens. The safety stocks are added at the third level which ensure the efficient use of the control policy. The method of developing the base-stock recovery policy was introduced in Gallego (1994).

It should be mentioned that an approximate decomposition approach is normally applied to such problems. Krieg and Kuhn decomposed a multi-product Kanban system (which is equivalent to a SELSP with lost sales) into multiple single product single-server vacation models. Thus, the individual subsystems can be evaluated numerically by an approximate continuous-time Markov chain. (Krieg and Kuhn 2002, and Krieg and Kuhn 2004)

Recently, Eisenstein (2005) extended the base-stock recovery policy developed by Gallego (1994). The new policy is more flexible and able to adjust the amount of idle time during recovery in response to the randomness.

Vuuren and Winands (2007) proposed an approximate decomposition approach to evaluate the quantity-limited lot-sizing policies. The quantity-limited lot-sizing policy is a policy which combines the sequence-cycle and base-stock strategies, that is, when the machine starts production of a product, it will continue production until either the base-stock level has been reached or a maximum number of items has been produced.

A classification scheme for the literature described in chapter III is shown in Table 3:

TABLE 3

An Overview of the Classification for Literature.

Policy Strategy	Reference
Sequence -cycle	Karmarkar and Yoo (1994), Qiu and Loulou (1995) Sox and Muckstadt (1997), Leachman and Gascon (1988), (1991) Fransoo (1992), Erkip et al. (2000) Markowitz et al. (2000), (2001), Bruin (2007)
Base-stock	Zipkin (1986), Winands et al. (2009) Altiok and Shiue (1994), (1995), (2000a), Brander et al. (2005), (2006) Paternina-Arboleda and Das (2005)
Combination of Sequence-cycle and Base-stock	Bourland and Yano (1994), (1994), Wagner and Smits (2004), (2004) Federgruen and Katalan (1996a), (1998), (1996b) Grasman et al. (2008), Gallego (1990), (1994) Eisenstein (2005), Krieg and Kuhn (2002), (2004) Vaughan (2003), Vuuren and Winands (2007)

CHAPTER IV

METHODOLOGY

IV.A Introduction of Approximate Dynamic Programming

Problems involving optimization over time, for example, the fleet scheduling, inventory management, portfolios investment, and asset selling problems, all involve making decisions during a time horizon based on the information obtained and the state status in each period. They are known as sequential decision problems (Powell, 2011). Many of them can be solved via dynamic programming using a backward recursion method. However, there are a large number of problems which suffer the curses of dimensionality and cannot be solved by the backward recursion method. In short, the curses of dimensionality arise when the dimensionality of the state, outcome or action space increases, which causes the volume of the space to increase very rapidly, which in turn causes the problem associated with the evaluation of the system under all the possible situations to be completely intractable. Approximate dynamic programming has been developed recently as a powerful tool to solve those kinds of problems.

Approximate dynamic programming arose as a solution technique for sequential decision problems involving uncertainty. Actually, the problem for

approximate dynamic programming was found to be independent from the problems for formal dynamic programming. Typically, the solution method of these problems can be referred to control theory, which estimates the parameters that control the system under a random environment. The development of approximate dynamic programming involved contributions from three domains: economics, operations research and computer science. At the very beginning, control theory was adopted by economists for problems involving control activities at a very basic level. The theory of controlling stochastic problems was mostly developed through the application of the theory of Markov Decision processes as associated with Bellman's work (Sutton and Barto, 1998). Computer scientists contributed their work by developing an algorithm. In the realm of artificial intelligence, it was found that the algorithm for reinforcement learning, an area of machine learning, is very suitable for solving approximate dynamic programming problems.

The theory of approximate dynamic programming was developed through the work of Bellman and Dreyfus since 1959, and the core theory of Markov decision processes resulted from Ronald A. Howard's work (Howard, 1960). The technique of approximate dynamic programming originated with Samuel's work (Samuel, 1959), within the artificial intelligence community. However, the benchmark where the technology of approximate dynamic programming was developed should be referred to the effort that combined control theory and neural network with the artificial intelligence in the 1990's, such as reinforcement learning and neuro-dynamic programming. Several books discussed these techniques in detail. For example,

Bertsekas and Tsitsiklis (1996), and Sutton and Barto (1998), as well as the edited volumes of Miller et al. (1990), and White and Sofge (1992). Two papers (Tsitsiklis 1994, and Jaakkola et al. 1994) merged dynamic programming and stochastic approximation theory, which enabled uncertainty to be formulated in dynamic programming problems. Later on, a series of papers (Godfrey and Powell. 2001, Papadaki and Powell 2003, Powell and Roy 2004, and Powell 2007) merged approximate dynamic programming with mathematical programming, which allowed approximate dynamic programming problems to be solved efficiently.

The general idea of approximate dynamic programming is based on an algorithmic strategy that steps forward through time instead of backwards through time as is typically done in the recursion method associated with formal dynamic programming problems. This "forward through time approach" will be introduced in detail in section IV.A.1.

IV.A.1 ADP Formulation and Algorithm

For a stochastic dynamic programming problem which has a planning horizon of T periods, the contribution for each period t is C_t. Now let $A_t^\pi(S_t)$ be a function that determines the decision given the information in the state variable S_t, where π is the policy chosen from the set of policies Π. The objective function for the stochastic optimization problem can be written as equation (1):

$$\max_{\pi} \mathbb{E}^{\pi}\{\sum_{t=0}^{T} \gamma^t C_t^{\pi}(S_t, A_t^{\pi}(S_t))\} \tag{1}$$

Note that γ^t is the discount factor for period t and $\mathbb{E}$ is the expectation associate with policy π in equation (1).

For many similar problems, solving equation (1) "as a single problem" might be computational intractable, however, it can be decomposed and solved by estimating the value associated with the system being in state S_t, which is represented by $V_t(S_t)$.

Let a_t be the decision being made at period t, and let S^M be a transition function which can determine the evolution of the system from the current state S_t to the next state S_{t+1}, associated with the decision a_t and the exogenous information W_{t+1} available between period t and $t+1$ as shown in equation (2):

$$S_{t+1} = S^M(S_t, a_t, W_{t+1}) \tag{2}$$

With the above transition function, it is possible to evaluate the value of being in state S_t, if decision a_t is taken. Thus, the best decision $a_t^*(S_t)$ for state S_t can be found by equation (3).

$$a_t^*(S_t) = \arg\max_{a_t \in \mathcal{A}_t}(C_t(S_t, a_t) + \gamma V_{t+1}(S_{t+1})) \tag{3}$$

Then, solving the problem is associated with estimating the value function (Bellman's Equation, Powell, 2008) of equation (4)

$$V_t(S_t) = \max_{a_t \in \mathcal{A}_t}(C_t(S_t, a_t) + \gamma V_{t+1}(S_{t+1}(S_t, a_t))) \tag{4}$$

As mentioned earlier, compared to the backward recursion strategy for the typical dynamic programming, approximate dynamic programming is based on a "step forward" algorithmic strategy. The value function $V_t(S_t)$ is estimated by going forward and following the sample path that is generated to simulate the evolution process of the system. There are many variations of approximate dynamic programming algorithms. Figure 4 only describes a basic procedure for the ADP algorithm.

Step 0. Initialization

Step 0a. Initialize $V_t^0, t \in \mathcal{T}$.

Step 0b. Set $n = 1$.

Step 0c. Initialize S_0^1.

Step 1. Choose a sample path ω^n.

Step 2. Do for $t = 0, 1, 2, ..., T$.

Step 2a. Solve:

$$\hat{V}_t = \max_{a_t \in \mathcal{A}_t^n} (C_t(S_t, a_t) + \gamma V_t^{n-1}(S^{M,a}(S_t^n, a_t))) \tag{5}$$

and let a_t^n be the value of a_t that solves (5).

Step 2b. If $t > 0$, update the value function:

$$\bar{V}_{t-1}^n \leftarrow U^V(\bar{V}_{t-1}^{n-1}, S_{t-1}^{x,n}, \hat{V}_t). \tag{6}$$

Step 2c. Update the states:

$$S_{t+1}^n = S^M(S_t^n, a_t^n, W_{t+1}(\omega^n)). \tag{7}$$

Step 3. Increment n. If $n \leq N$ go to Step 1.

Step 4. Return the value functions $(\bar{V}_t^N)_{t=1}^T$.

Figure 4. A Basic ADP Algorithm

IV.A.2 Elements in ADP

There are five elements to a dynamic programming model, as well as to an approximate dynamic programming model:

- State Variable. The state variable is the most important quantity in an approximate dynamic programming model. It captures the current status of the system. And, it is necessary as input to compute the decision function value, the transition function value, and the objective function value, and thus, constructing the value function approximation.

- Decision Variable. Decision variable is generated through the rule, policy, strategy or function which is used to make the decision under a particular circumstance. In the Markov decision process, this decision variable is called an action, and is denoted by $a \in \mathcal{A}$. In the optimal control community, this decision variable is called a control, and is denoted by $u \in \mathcal{U}$.

- Exogenous Information Process. The arrival of exogenous information mainly represents the uncertainty in the system, which is the exogenous factor that changes the state of the system. How to deal with the exogenous information processes and make decisions before they arrive is the central challenge of approximate dynamic programming problem.

- Transition Function. The evolution of the system from current state to the next state is specified by the transition function, which incorporates the effect

of the decision and exogenous information to the system.

- Objective Function. In an approximate dynamic programming problem, the objective function is what to be optimized through developing a better policy. It is called a contribution function for a maximizing problem and a cost function for an minimizing problem.

IV.A.3 Perspectives of ADP

Powell (2007) discussed the wide range of promising areas to which approximate dynamic programming could be applied, including the areas of transportation, inventory control, finance, energy, military, manufacture and medical. He also mentioned that approximate dynamic programming is typically very suitable for solving complex dynamic programming problems which cannot be handled by the backward recursion method (Powell, 2008) ; finally, Powell mentioned that there are three main perspectives associated with approximate dynamic programming:

- Large-scale (deterministic) optimization. Simio et al. (2009), Topaloglu and Powell (2005) had applied approximate dynamic programming to solve a large-scale resource allocation problem in transportation. From their research, ADP was found to be not only a tool to handle the uncertainty, but also a decomposition strategy that breaks problems with long horizons into a series of shorter horizon problems.

- Making simulations intelligent. There are many stochastic, dynamic problems that are solved using myopic policies to make decisions at the current time without considering the impact on the future. Approximate dynamic programming can make the future tractable, thus providing higher quality decisions and therefore a more intelligent approach. Powell et al. (2012) combined ADP with a simulation model and solved an energy allocation problem more intelligently.

- Solving complex dynamic programs. The uncertainties in dynamic programs always complicate the problem. However, the various algorithms and approximation strategies belonging to approximate dynamic programming often provide good solutions to these intractable stochastic dynamic programming problems. Topaloglu and Powell (2007) coordinated decisions on pricing with a stochastic dynamic freight carrier system, then solved their intractable complex dynamic program via ADP and obtained high-quality solutions.

Many problems involving complex and dynamic production systems are solved by simulation modeling, since users do not have to formulate a explicit mathematical model and thus avoid modeling the complexity of dynamics in the systems. The literature involving the application of approximate dynamic programming to such problems is relatively rare. There are only two papers which have applied approximate dynamic programming to the stochastic economic lot size

problem. Paternina-Arboleda and Das (2005) applied reinforcement learning with a simulation to analyze a base stock policy. However, the policy they developed by the use of an artificial neural-network is very difficult to implement, thus the result is not readily known. Lhndorf and Minner (2012) modeled the SELSP by semi-Markov decision processes and solved it by approximate value iterations (AVI) with gradient search. They also compared the results from ADP to those generated from a direct policy search via simulation, and found that the classic ADP approach of AVI and stochastic gradient updates is not competitive for larger problems. Lhndorf and Minner (2012) pointed out that the SELSP would be a good benchmark for future research in ADP.

As mentioned in section IV.A.3, ADP is very suitable for solving problems involving complex and dynamic systems by allowing the simulation to operate a more intelligent fashion. Although applying a basic approximate value iteration to SELSP is not very competitive when compared to a policy found by direct search via simulation as shown in Lhndorf and Minner's paper (Lhndorf and Minner, 2012), the various classes of approximation schemes and the richness of learning algorithms in ADP are still worthy to try.

IV.B Policies in ADP

A policy is very essential in an approximate dynamic programming model, since it defines the rule by which the system makes decisions at a certain state with available exogenous information.

Any method for determining an action at a given state can be viewed as a policy. Thus, the policy might be a simple look-up table, a function or even a complex algorithmic strategy. There are four categories of policies associated with approximate dynamic programming:

- Myopic Policies. Myopic policies are the most easily understandable class of policies. They do not require the use of future information or any forecasting technique. The decisions are only based on the current status of state S_t, which can be represented as equation (8):

$$A^{Myopic}(S_t) = \arg\max_{a} C(S_t, a) \tag{8}$$

  The form of a myopic policy would be any mathematical programming model which only considers the status of the current state.

- Lookahead Policies. Lookahead policies make decisions by considering the information over some horizon. These policies commonly solve problems by using the approximation of future information over a limited horizon to choose actions. For example, the tree search, roll-out heuristics, and predictive control are all methods for lookahead policies.

- Policy Function Approximation. Some systems may have the feature that their processes associated with making decisions can be easily captured by a function without any embedded optimization problem. An example of this type of policy would be selling a stock when its price goes over μ. The method

of developing this kind of function is called a policy function approximation. To approximate the policy function, the simplest way to generate a look-up table; the most widely used method is to design a parameterized function to generalize the policy, like the form of $A^{\pi}(S_t|\theta)$; moreover, the non-parametric statistical method can be used to fit the function and return the action a, such as kernel regression.

- Value Function Approximation. Value function approximation is considered the most powerful tool for solving complex dynamic programming problems. This approach uses the value function $V_t(S_t)$ to approximate the system value of being in a certain state, and thus avoids "the curse of dimensionality". The decision can be made through solving the value function as equation (9):

$$a_t^*(S_t) = \arg\max_a (C(S_t, a) + \bar{V}(S^{M,a}(S_t, a))) \tag{9}$$

Since the part of $\bar{V}(S^{M,a}(S_t, a))$ in equation (9) does not only capture the current status of the system but also the future information, this approximation does consider the impact of making a decision now on the future and has the potential to generate a better solution. There are three strategies to approximate the value function: look-up table, parametric model and non-parametric model.

The myopic policies, lookahead policies, policy function approximation and value function approximation constitute the core tools for solving ADP problems. However, hybrid strategies can also be generated based on these approaches. For

example, predictive control can be used with value function approximation, roll-out heuristics can be associated with policy function approximation, and tree search can be combined with roll-out heuristics and a look-up table policy.

Since a revising scheme is needed to accurately approximate the value of being in a state, an exploration versus exploitation issue may arise when applying certain algorithms to search for an optimal policy. That is: if an policy involves on exploiting current estimates of downstream values which are thought to be the best possible decision, it may miss the chance to visit some states where better solutions exist. Therefore, a randomized policy may need to be used in the algorithm and works with the other policies ("exploitation" policies). The randomized policies work by randomly choosing an action rather than following the "exploitation" policies, thus it can help the algorithm explore more state and action then learn their values. The most widely used randomized policy is an ϵ-greedy policy. In this policy, an action $a \in \mathcal{A}$ will be chosen at random with probability ϵ, and with probability $1 - \epsilon$ it will follow the "exploitation" policies.

IV.C Value Function Approximation Scheme

Value functions can be used to approximate the system value of being in a state S_t, thus generating a policy based on the value function $\bar{V}_t(S_t)$ and helping to make decisions under certain circumstances through equation (10):

$$a_t^*(S_t) = \arg\max_{a_t \in \mathcal{A}_t}(C_t(S_t, a_t) + \gamma \mathbb{E}\bar{V}_{t+1}(S_{t+1})) \tag{10}$$

Therefore, value function approximation is considered as the most powerful tool in approximate dynamic programming. However, the way to develop a good approximation model and the method to iteratively learn the function is a challenge for researchers.

In section IV.C, several typical methods for approximating the value function is introduced. The methodology of learning the function value will be discussed in chapter VI.

IV.C.1 Aggregation

In the very early stages of research for approximate dynamic programming, aggregation was used to reduce the size of the state space thus overcoming the "curse of dimensionality". The idea was to firstly aggregate the multiple dimensions in the original problem. After solving the problem on the aggregated level, an approximate solution will be disaggregated to the original problem. (Powell, 2011)

When aggregation is used to approximate the value function, the objective function for the problem will become the form of equation (11):

$$\max_{a_t \in \mathcal{A}} (C_t(S_t, a_t) + \gamma \mathbb{E} \bar{V}_{t+1}(G(S_{t+1}))) \tag{11}$$

It can be seen that the original value function $\bar{V}_{t+1}(S_{t+1})$ is replaced by $\bar{V}_{t+1}(G(S_{t+1}))$ in equation (11), where $G(S_{t+1})$ is an aggregation function that simplifies the expression of original state S_{t+1} by ignoring the dimensions, discretizing it, or applying any other method to reduce the state space. This would

also reduce the number of parameters need to be estimated.

The most widely used aggregation method is hierarchical classification, since the hierarchical aspect is very natural in common world. For example, a portfolio problem may need to estimate the values of investing money in the stocks of many particular companies. It might be a good way to aggregate companies by their industry segments (e.g., electronic, chemical and service). For each industry, it could be further aggregated based on whether the company is viewed as domestic or multinational, and so on.

IV.C.2 Parametric Models

Using aggregation is still a form of look-up table. Although using aggregation allows the avoidance of exploiting the huge number of possible values for state vectors, it does not consider the specialized structure in the state variable, which may lead to some level of inaccuracy.

Using parametric models to approximate the value function could not only reduce potential large state variables but also keep the structure in the state variables. Linear regression modeling (see equation (12)) is the most widely used parametric model in ADP to approximate the value function:

$$y = \theta_0 + \sum_{i=1}^{I} \theta_i x_i + \epsilon \tag{12}$$

Equation (12) is the classical representation of the linear regression model, where θ are the parameters to fit the prediction variables (responses) y (which

would be a set of observations), and ϵ is a random error.

If we consider θ to be the column vector of parameters, and y as the column vector of responses, we can write equation (12) as:

$$y = \theta^T x + \epsilon \tag{13}$$

Note that ϵ is a vector of errors $(\epsilon_1, ..., \epsilon_n)$, which we assume to be independent and identically distributed. The format of the linear regression model in equation (13) can be addressed by using Approximate Dynamic Programming more easily.

Support vector machine and support vector regression (Powell, 2011) are also very popular parametric models in Approximate Dynamic Programming for approximating value functions. As we know, support vector machine is used to address classification problem (for example, if we have discrete function value), while support vector regression is used to fit continuous functions.

The method associated with support vector machine (regression) is based on the idea of using an optimal hyperplane to differentiate the classes of data points by maximizing the margin between the classes, as Figure 5 shows:

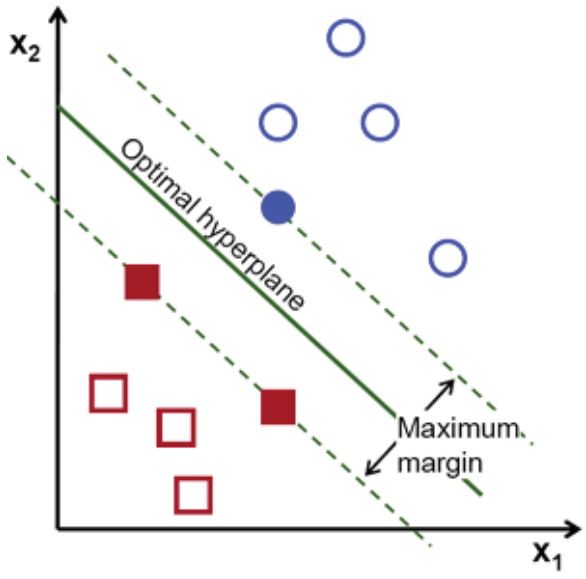

Figure 5. Support Vectors Scheme.

Note that $y = wx + b$ is the mathematical representation of the hyperplane, ϵ is the distance from the edge of one of the classes to the hyperplane, and the margin we want to maximize would be $\frac{2\epsilon}{||w||}$. Therefore, the optimal hyperplane can be obtained by solving the following optimization model (normally through Lanrange relaxation):

$$\min \frac{1}{2}||w||^2 \tag{14}$$

$$\begin{aligned} \text{s.t. } & y_i - wx_i - b \leq \epsilon \\ & wx_i + b - y_i \leq \epsilon \end{aligned} \tag{15}$$

IV.C.3 Non-parametric Models

In approximate dynamic programming, the parametric models approximation scheme is very powerful for obviating the problem of estimating the huge number of state variable, as well as keep the structure in the problem. The parametric models is also effective, since it is relatively easy to solve. However, the design of an effective parametric model is considered as a frustrating art. For many problems, it is very difficult to develop a parametric model that works well. For this reason, non-parametric statistical methods have attracted more attentions in recent years. (Powell, 2011)

Compared to parametric models, non-parametric models work primarily by building local approximations to functions using observations rather than depending on functional approximations, which can avoid the difficulty of designing a parametric model suit to the problem.

The k-nearest neighbor is a non-parametric method for classification and regression. In this method, the functions is estimated by using a weighted average of the nearest neighbor points. The model is given by equation (16):

$$\bar{Y}^n(x) = \frac{1}{k} \sum_{n \in \mathcal{N}^n(x)} y^n \tag{16}$$

In equation (16) y^n is assumed to be a response to measure a distance metric between a query point x and an observation x^n. $\mathcal{N}^n(x)$ is the set of the k-nearest points to the query point x. Therefore, $\bar{Y}^n(x)$ is the estimate of true function $Y(x)$

given the observations $x^1, ..., x^n$.

Another non-parametric model named as kernel regression can also be applied in approximate dynamic programming. Compared to the k-nearest neighbor, kernel regression uses a weighted sum of prior observations to estimate $\bar{Y}^n(x)$. The model can be generalized as equation (17):

$$\bar{Y}^n(x) = \frac{\sum_{m=1}^{n} K_h(x, x^m) y^m}{\sum_{m=1}^{n} K_h(x, x^m)} \tag{17}$$

In equation (17), $K_h(x, x^m)$ is a weighting function that decreases with the distance between the query point x and the measurement x^m. h is referred to as the bandwidth which plays the role of scaling.

A large group of non-parametric statistical computational models called artificial neural networks have been widely used in approximate dynamic programming. Artificial neural networks arose from the field of computer science and is inspired by the animal central nervous systems. It is very suitable to address machine learning and pattern recognition problems. We will discuss detail in section VI.C.1.

IV.D A Simple Example for ADP(Gosavi, 2009)

Section IV.C has discussed the various schemes to approximate the value of being in a certain system state. After a specific approximation scheme is chosen, one still need an algorithm to compute the approximated state value $\hat{v}^n$, and update the parameters in the model of the approximation scheme, thus learn the configurations

of the approximation. There are many algorithms available that can be chosen from the field of machine learning. We will discuss the algorithms we adopted to solve the problem in chapter VI. At the end of chapter IV, we will provide a simple example to illustrate how Approximate Dynamic Programming works.

Consider a two-state Markov Decision Process (MDP) in which two actions are permitted in each state. The relevant data are supplied in Figure 6. The example illustrates the nature of a generic MDP. Theoretically speaking, underlying any MDP is data with a structure similar to this two-state MDP; for large-scale MDPs, usually the transition probabilities cannot be determined easily. However, simple examples such as these can serve as test-beds for numerically testing a newly designed RL algorithm.

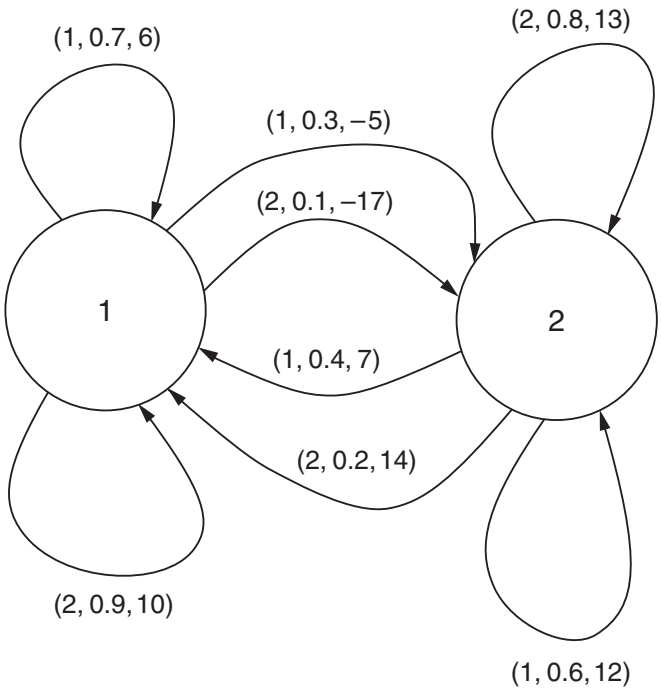

Figure 6. Two-State MDP with (x, y, z) on Each Arc Denoting the Action x, Transition Probability y, and Immediate Reward z Associated with the Transition.

CHAPTER V

MODEL FORMULATION

In chapter V, a Markov decision process model is proposed for the problem discussed in section II.B. Consider a production system which is producing multiple finished products and intermediate components with several work centers. The demands of these finished products are uncertain. For each work center, significant changeover time is required for setting up to produce a certain item. The production process is divided into two stages by the intermediate components. Hence, there would be two inventory pools which would need to be controlled for the system, one for the intermediate components and the other for the finished products. A finished product cannot be produced if lacks the related type of intermediate component.

V.A Notations

The model assumes that there are N total products, including I finished products and J intermediate components ($I + J = N$). The work centers K can each only produce one item at a time. If a work center production status changes from idle to busy or from producing one type of item to producing another, a significant setup time T_n ($n \in I \cup J$) is required. The time to process one unit of

item on work center k is assumed to be deterministic and to have the same value for each type of item. The work center cannot be interrupted when performing a changeover or during the production of a single item. The production rate for work center k is fixed to Pr_k. The inventory holding cost per unit time for item n is h_n ($n \in I \cup J$). Customer demand which is not met at the time of demand for finished product i is assumed to be backordered with a cost of b_i per unit time. The inventory level for intermediate product j is Inv_j and the inventory level for finished product i is Inv_i. Inventory level for any product n ($n \in I \cup J$) during unit time period t is Inv_{nt}. Unit changeover cost for product n is U_n, which can be assume to be 1 in our problem. The state of work center k is M_k, which indicates the current working status of work center k. For example, if $M_k = 0$, work center k is idle, and if $M_k = 1$, work center k is producing product 1. F_k is the set of potential states associated with work center k. For example, if work center k can only produce product 1, 3 and 6, we would have $F_k = \{0, 1, 3, 6\}$; note that 0 means the work center is idle. The actions we can take at work center k are denoted by a_k. If $M_k = a_k$, the status for work center k is not changing; if $M_k \neq a_k$ and $a_k \neq 0$, the changeover will be performed on work center k; if $M_k \neq a_k$ and $a_k = 0$, work center k will be set to idle in the next period. t is the unit time period, which is the smallest time unit in our simulation, while τ is the decision period which might be constituted by multiple t. The total cost during decision period τ is C_τ, which we want to minimize.

The notations used in the model is shown in Table 4:

TABLE 4

Table of Notations.

$i \in I$:	Index for finished product.
$j \in J$:	Index for intermediate component.
$n \in N$:	Index for finished product and intermediate component, $N = I \cup J$.
$k \in K$:	Index for work center.
f_k:	Index for status which can be performed on work center k, $f_k \in F_k$.
F_k:	Set for potential status can be performed on work center k.
Inv_n:	Inventory level for product n.
Inv_{nt}:	Inventory level for for product n during unit time period t.
T_n:	Time for setting up for item n.
h_n:	Holding cost for product n per unit time.
U_n:	Unit cost for product n associated with changeover.
b_i:	Backorder cost for finished product i per unit time.
Pr_k:	Production rate for work center k.
M_k:	State of work center k.
a_k:	Action made on work center k.
t:	Unit time period.
τ:	Decision period.
C_τ:	Total cost during decision period τ.

V.B Assumptions

V.B.1 State $\boldsymbol{S}$ and Action $\boldsymbol{a}$

The overall state of the system $\boldsymbol{S}$ is defined by the work center status M_k for each work center k and the inventory level Inv_n for each item n:

$$\boldsymbol{S} = (M_1, ..., M_K; Inv_1, ..., Inv_N)$$

S_k is the state of a single work center k, which is defined by M_k and the inventory level Inv_{f_k} ($f_k \in F_k$) for each item which is associated with work center k:

$$S_k = (M_k; Inv_{1_k}, ..., Inv_{F_k})$$

Note that each item is restricted to a particular location for its inventory. When a decision is made to produce item f_k on work center k, the value of f_k will be assigned to the action a_k ($a_k = f_k \in F_k$) on work center k ($a_k = 0$ means to make work center k idle). All the possible actions for the K work centers consist of the action space for the system:

$$\boldsymbol{a} = (a_1, ..., a_K)$$

Note that the state and action spaces in this problem suffer from the "curse of dimensionality". If the state variable $S = (M_1, ..., M_K; Inv_1, ..., Inv_N)$ has K dimensions of machine status and N dimensions of inventory, and the possible value for each machine status M_k is R and the possible value for each item's inventory Inv_n is L, then we have $R^K L^N$ different states for the system. In a similar sense, if the possible value for each action a_k is D, we have D^K action space for K dimensions of actions in the system. Therefore, it may very well be computationally infeasible to evaluate the qualities of all the pairs of state and action value $(\boldsymbol{S}, \boldsymbol{a})$,

even for a fairly small size problem. For example, the number of possible $(\boldsymbol{S}, \boldsymbol{a})$ is $3^3100^83^3$ for a $R = 3$, $L = 100$, $D = 3$, $K = 3$, $N = 8$ problem. However, approximate dynamic programming is an efficient method to avoid the curse of dimensionality and make the evaluation of $(\boldsymbol{S}, \boldsymbol{a})$ possible.

V.B.2 Demand Arrivals

In the system, the demands only arrive for the end products. Although the work centers that produce the end products also require items from the intermediate inventory pool, we consider the requirement as inner control variables in the system.

The demands for each end product i are modeled as compound Poisson processes through a compound Poisson distribution, which is the probability distribution of the sum of a number of independent identically-distributed random variables, where the number of terms to be added is itself a Poisson-distributed variable. In contrast to a pure Poisson process, the compound Poisson processes can model a stochastic demand process where the variance is different from the demand rate. A compound Poisson process corresponds to a continuous-time stochastic process in which arrivals follow a Poisson distribution $P_i^A(k)$ and demand sizes per arrival follow a geometric distribution P_i^D. The arrival times and demand sizes are assumed to be independent for each end product.

Equation (18) represents the probability for the arrivals of demand for the finished products being equal to k during a time interval of length t. λ_i is denoted as the demand rate of demand for product i, with mean number of arrivals $\lambda_i t$.

$$P_i^A(k) = \frac{(\lambda_i t)^k}{k!} \exp(-\lambda_i t) \tag{18}$$

Equation (19) represents the probability of the demand size being equal to d for each arrival with mean demand size $1/q_i$. q_i is the parameter of the geometric distribution used, where $0 < q_i \leq 1$.

$$P_i^D(d) = q_i(1 - q_i)^{d-1} \tag{19}$$

V.B.3 Cost Function

The cost associated with operating the system arises from three sources: the holding cost, the backorder cost, and the setup cost. The setup cost is measured through the production rate Pr_k. During setup times or wait times for setups at work center k, nothing is produced at that work center.

The cost function $C_\tau(\boldsymbol{S}, \boldsymbol{a})$ for each decision period τ can be represented as equation (20):

$$C_\tau(\boldsymbol{S}, \boldsymbol{a}) = \begin{cases} \sum_t^\tau \sum_n^N h_n \max(0, Inv_{nt}) - \sum_t^\tau \sum_i^I b_i \min(0, Inv_{it}), \text{ if no changeover} \\ \sum_t^\tau \sum_n^N h_n \max(0, Inv_{nt}) - \sum_t^\tau \sum_i^I b_i \min(0, Inv_{it}) + \sum_k^F U_n T_n Pr_k, \text{ o.w.} \end{cases} \tag{20}$$

The decision period τ is the time between two decisions, while t represents unit time ($t \in \tau$). Whenever any machine produces one unit of any item, a decision is required. For instance, τ would be $\frac{1}{Pr_k}$ if no changeover is performed on machine

k during the decision period (τ is equal to the time that one unit of item is produced by machine k), or τ would be $T_n + \frac{1}{Pr_k}$ if changeover to item n is performed on machine k during the decision period (note T_n is the setup time for item n). If machine k is set to be idle, the decision epoch time τ would be from current point to the next arrival of any demand on machine k. When the new demand arrives, a decision will be made on the idle machine.

The cost function $C_\tau(\boldsymbol{S}, \boldsymbol{a})$ calculate the cost for the production system during decision period τ by two conditions:

- If there is no changeover during the decision period τ, $C_\tau(\boldsymbol{S}, \boldsymbol{a})$ would only consider the holding cost of N items in this system ($\sum_t^\tau \sum_n^N h_n \max(0, Inv_{nt})$) and the backorder cost for I finished product ($-\sum_t^\tau \sum_i^I b_i \min(0, Inv_{it})$).

- If there is any changeover during the decision period τ, $C_\tau(\boldsymbol{S}, \boldsymbol{a})$ would consider the holding cost of N items and the backorder cost for I finished product, plus the changeover cost $\sum_k^F U_n T_n Pr_k$.

Note that $\max(0, Inv_{nt})$ will always keep the inventory level for product n as positive to calculate the holding cost, and $\min(0, Inv_{it})$ will always keep the inventory level for finished product i as negative to calculate the backorder cost.

CHAPTER VI

CONTROL POLICIES AND ALGORITHMS

VI.A Optimization via Simulation by Adjusted (s, S) Policy

VI.A.1 Optimization via Simulation

As we mentioned in section II.B.3, production systems such as the GE production system, are difficult to represent as an analytical model, because of the uncertainties and the complicated interactions in the systems. To represent the system and to perform analysis and improvement, simulation is an appropriate tool.

Many researchers have demonstrated the benefits of simulation for modeling and analyzing complex production systems (Souza et al., 1996, Lin et al., 1998, Benedettini and Tjahjono, 2009). Benedettini and Tjahjono (2009) have also pointed out that the complexities of dynamics in production systems can be explicitly reproduced by simulation models. These complex inventory policies for operating the production systems, such as CONWIP (Huang et al., 2007), AWIP (Masin and Prabhu, 2009) and (s, S) inventory policy (Hu et al., 1993), are evaluated through simulation models. Also, optimization via simulation is an excellent tool for comparing different system configurations, and is even able to

improve the production systems or obtain optimal control variable values (Nyen et al., 2006; Kumar and Sridharan, 2007; Han and Zhou, 2010).

VI.A.2 Adjusted (s, S) Inventory Control Policy

Inventory which includes raw materials, work-in-process components and finished goods, is often used as a primary control variable in design of production strategies for a production systems. For a multi-product manufacturing system, the control policy must be able to determine and inform the work stations when to stop producing the current product and switch to another product to produce. Altiok and Shiue (2000b) propose a continuous-review (R, r) policy for controlling a pull-type production system with multiple product types. The production of a particular product stops when its inventory level reaches its target value R, and a request to initiate the production of a product is made as soon as its inventory level drops to or below its reorder point r. The (R, r) policy is very similar to the (s, S) policy developed by Scarf (1959) which is very widely used. An (s, S) policy in a manufacturing system will consider production stops at the instant that the inventory level is raised to S, while production begins again at a review point when the inventory level is observed to have dropped to or below s for the first time (Lee and Srinivasan, April 1988). The review point can be either continuous or discrete. The (s, S) policy is known to be effective in a variety of inventory situations (Veinott, 1967).

The challenge to apply a classic (s, S) policy on the problem in our research

is: if there are two products associated with one machine whose inventory levels both dropped below s, a conflict of which product to produce will arise. To deal with this conflict, we proposed an adjusted (s, S) inventory policy to solve this problem in an simulation model. In the adjusted (s, S) inventory policy, a trigger variable P^* is added into the classic (s, S) policy. Two definitions are needed before introducing the trigger variable P^* (Wu et al., 2013).

The production system can be generalized with J types of intermediate components (e.g., $j = 1...9$ for the fabricated racks in GE problem), I types of products (e.g., $i = 1...13$ for the coated racks in GE problem) and K work centers (e.g., $k = 1...5$ for the five work centers in GE problem). Since these work centers are operating separately, the first definition is given below:

Definition For a work center k, the set for the types of intermediate components or products associated with it can be denoted as S_k, where $S_k \in I$ or J.

To design an inventory policy for controlling a multi-product production system, one must define the decision variables which define when to stop producing a certain product and which product is needed to be replenished. Based on the concept of the (s, S) policy, the stopping criterion is developed as the second definition:

Definition A work center k will stop producing product n when the inventory level of product i (denoted as Inv_n) reaches its S_n.

As soon as the production of product n is stopped, a changeover must be performed, and a replenishment product m^* must be picked up from S_k. The index of m^* is determined by a heuristic trigger variable P^*, which is given by equation (21):

$$P^* = \min\{m \in S_k | P_m = \frac{Inv_m - s_m}{s_m}\} \tag{21}$$

In (21), P_m actually represents the negative portion of the interval that product m's current inventory level differs from its s_m in (s, S) policy, and P^* is the minimum value of P_m one can find among the set of S_k. The product index which needs to be replenished (denoted as m^*) is the index of m associated with P^* at work center k. Then m^* can be computed by (22):

$$m^* = \arg\min_{m \in S_k} P_m \tag{22}$$

To apply the adjusted (s, S) inventory policy we proposed, a pair of control parameters (s_n, S_n) is assigned to each product n to control its inventory. As soon as product n's inventory level reaches to S_n, the machine will stop to produce product n and a non-production period for product n begins. During the non-production period for product n, the system will check the inventory level for product n every time when the work center stops. If the inventory level for product n is observed to be at or below s_n, and the trigger variable P^* also point to n, then the machine will start to produce product n and a new production period for

product n begins (refer to figure 7).

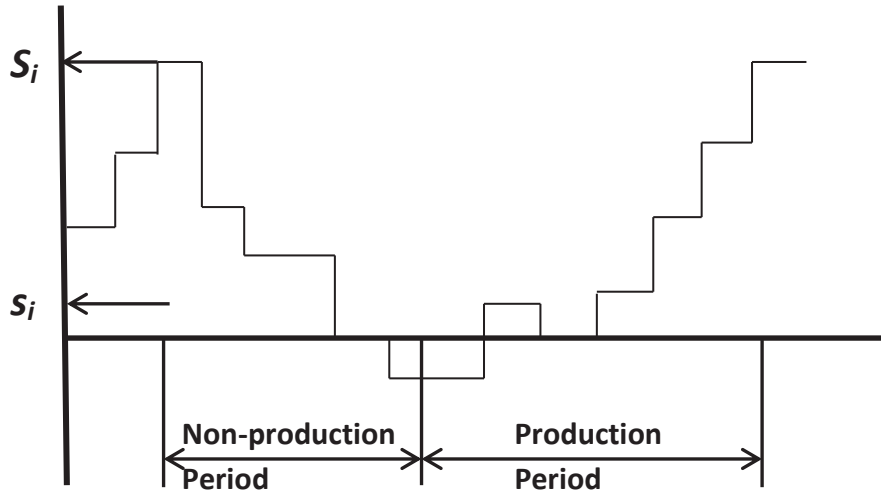

Figure 7. The (s, S) policy with trigger variable.

VI.B Linear Approximation with Stochastic Gradient Search

VI.B.1 Linear Regression with Basis function

As we mentioned in section IV.C.2, a linear regression model is a good parametric model to approximate the value function in Approximate Dynamic Programming.

When implementing a linear regression model to ADP, a basis function $\phi_f(S)$ is often used to replace the independent variable x_i. $\phi_f(S)$ might be an indicator variable, a discrete number, or a continuous quantity. A basis function plays the role of extracting information from the state variable S and helps to explain the behavior of the value function. With the linear regression model and basis functions, the value function in ADP can be written in the form of equation (23):

$$\bar{V}(S|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S) \tag{23}$$

where θ_f is the parameter vector in equation (23).

At this point, the mission to solve the approximate dynamic programming problem is to find the appropriate parameter vector of θ_f. Consider the general version of a linear regression model with basis functions as equation (24):

$$Y = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S) + \epsilon \tag{24}$$

where Y is the observation of the value of being in a state, and ϵ explains any error associated with the difference between the observed value and the regression estimation. Then the parameter vector of θ_f can be solved by the stochastic gradient search method which is introduced in section VI.B.3.

Note that the linear regression model with basis functions often performs as a good approximation scheme in ADP and is relatively easy to solve. However, there are many problems which exhibit nonlinear behavior, then some other parametric models should be applied to approximate the value function rather than the linear regression model. For example, a quadratic polynomial such as equation (25) might be used as the strategy to approximate the value function:

$$\bar{V}(S) = \sum_{i} (\theta_{1i} S_{ti} + \theta_{2i} S_{ti}^2) \tag{25}$$

VI.B.2 Linear Regression Approximation Scheme

To solve the problem, a set of value functions are used to approximate the average costs for the next decision period by taking certain action, as equation (26) shows:

$$\bar{V}_{(a_1,...,a_K)} = \sum_{k=1}^{K} \theta_k M_k + \sum_{i=1}^{I} \theta_{+i} \max(0, Inv_i) + \sum_{i=1}^{I} \theta_{-i} \min(0, Inv_i) + \sum_{j=1}^{J} \theta_j Inv_j \quad (26)$$

The set of value functions $\bar{V}_{(a_1,...,a_K)}$ is actually a list of linear regression models with basis functions which are defined by the combination of the actions $\boldsymbol{a} = (a_1, ..., a_K)$. For different combinations of actions, the value function $\bar{V}_{(a_1,...,a_K)}$ is different with respect to their parameter vector $\theta = (\theta_k, \theta_{+i}, \theta_{-i}, \theta_j)$. When a decision is required to be made in the system, all the response values $\bar{V}_{(a_1,...,a_K)}$ in the list of linear regression models will be calculated from the current system state value $\boldsymbol{S} = (M_1, ..., M_K; Inv_1, ..., Inv_n)$ with respect to each combination of action $\boldsymbol{a} = (a_1, ..., a_K)$. Then, the optimal action $\boldsymbol{a}^* = (a_1^*, ..., a_K^*)$ will be chosen as the minimum value of $\bar{V}_{(a_1,...,a_K)}$ found in the value function list by equation (27):

$$\arg\min_{a^* \in \mathcal{A}} \bar{V} \quad (27)$$

note that A is the action space for the system.

In summary, we are using a set of linear regression models to evaluate the average cost $\bar{V}$ for making a certain action $\boldsymbol{a}$ under system state $\boldsymbol{S}$. The number of the linear regression models in the value function set depends on the action space A.

If there is only a single machine in the production system, say machine 1, we will have the action space $A = a_{machine1}$. That is, the action space A only depends on the possible status of machine 1. If machine 1 only has 3 possible states, the action space A is equal to 3. Thus the number of linear regression models in the value function is set to 3. However, in our research, we are trying to address the problem for multiple machines in the production system. Therefore, the action space A depends on the combination of the actions taken on each machine, where $A = \{a_1, a_2, ..., a_n\}$ if there are n machines in the production system. Hence, the action space is equal to 3^n, if all the machines in the production system only has 3 possible states, and our linear regression approximation scheme still suffers from the "curse of dimensionality" with respect to the number of machines in the production system, although it is still a good approximation method for a production system which only has a few machines.

VI.B.3 Stochastic Gradient Search

A stochastic gradient algorithm is a popular method which is very suitable for using an approximate value iteration to update the value function. (Bertsekas, 2007) It has the advantage that the estimate of a parameter can be updated online while new samples are being collected.

A general stochastic optimization problem follows the form of (28):

$$\min_x \mathbb{E}F(x, W) \tag{28}$$

where the W contains the random information in the problem.

Due to the stochastic characteristics of W, the gradient of the decision variable x cannot be computed exactly by the derivative of equation (28) as a deterministic optimization problem. However, for many problems, the random information can be fixed by following a sample realization w as $W = W(w)$. After that, the gradient of the new objective function $F(x, W(w))$ can be found by derivative. And the decision variable x can be updated by following equation (29):

$$x^n = x^{n-1} - \alpha_{n-1} \nabla_x F(x^{n-1}, W^n) \tag{29}$$

In equation (29), $\nabla_x F(x^{n-1}, W^n)$ is called a stochastic gradient since it depends on a sample realization of W^n, and α_{n-1} is a stepsize.

Consider that a linear value function approximation with basis functions is already set up by $\bar{V}(S|\theta) = \sum \theta \phi(S)$, as it is introduced in section IV.C.2. To find the best value of the parameter θ, one needs to solve equation (30) which involves the minimization of the mean squared error:

$$\min_{\theta} \mathbb{E} \frac{1}{2} (\bar{V}(S|\theta) - \hat{v}^n)^2 \tag{30}$$

After applying the stochastic gradient algorithm, the updating step of θ will be:

$$\theta^n = \theta^{n-1} - \alpha_{n-1} (\bar{V}(S|\theta^{n-1}) - \hat{v}^n) \nabla_\theta \bar{V}(S|\theta^n) \tag{31}$$

Since $\bar{V}(S|\theta^n) = (\theta^n)^T\phi(s)$, the gradient with respect to θ is given by $\nabla_\theta \bar{V}(S|\theta^n) = \phi(s^n)$, where $\phi(s^n)$ is the sample realization at iteration n. Then equation (31) can be updated to:

$$\theta^n = \theta^{n-1} - \alpha_{n-1}(\bar{V}(S|\theta^{n-1}) - \hat{v}^n)\phi(s^n) \tag{32}$$

For applying the stochastic gradient algorithm, an initial estimate of the parameter θ^0 is required, and it is usually set to $\theta^0 = 0$.

VI.B.4 ADP Algorithm: Stochastic Gradient Search by Approximate Value Iteration

To implement the stochastic gradient search within an approximate dynamic programming algorithm by following equation (32), we use discounted reward to represent the n^{th} observation value $\hat{v}^n$ which is generated by the sample path ω^n, as Equation (33) shows:

$$\hat{v}^n = C_n(S_n, \boldsymbol{a}_n) + \exp(-\gamma\tau) \min_{a_n \in \mathcal{A}} \bar{V}^{n-1}(S_n, \boldsymbol{a}_n; \theta_n) \tag{33}$$

In Equation (33), $C_n(S_n, \boldsymbol{a}_n)$ is the immediate cost at period n, which is returned by the sample realization. $\bar{V}^{n-1}(S_{n+1}, \boldsymbol{a}_n; \theta_n)$ is the estimation of the value of being in successor state S_n by using the approximation model updated in iteration $n-1$. $\gamma \in (0, 1)$ is a discount factor related to the algorithm.

As it is introduced in section VI.B.3, the stepsize of α_{n-1} is another very important parameter which need for the stochastic gradient search. The stepsize

will largely affect the convergence speed when the algorithm is performing, however, too large of a stepsize might significantly reduce the solution quality. Although the optimal value of the stepsize is unknown for the stochastic gradient algorithm, experimental work has shown that there exist a simple stepsize rule that work well in practice. (Powell, 2011) The rule to determine the stepsize is given by:

$$\alpha_n = ab(a + n - 1)^{-1} \tag{34}$$

where $a \in R^+$ and $b \in (0, 1]$ are two scaling parameters in the algorithm.

An approximate value iteration algorithm (shown in figure 8) is proposed to solve the problem.

Step 0. Initialization: $\theta^{\mathbf{0}}$, starting state S^0.

Step 1. Generate a sample path.

Step 2. Do for $n = 1, 2, ...$

Step 2.1 Find action a^* by solving:

$$\arg \min_{a_n^* \in \mathcal{A}} \bar{V}(S_n, a_n; \theta_{\mathbf{n}}). \tag{35}$$

Step 2.2 Obtain information for decision period τ by following sample realization ω:

$$(C_n(S_n, \boldsymbol{a}_n), \tau_n, S_{n+1}) \leftarrow S^M(S_n, a^*, \omega) \tag{36}$$

Step 2.3 Compute $\hat{v}^n$ through equation (33).

Step 2.4 Update $\theta_{\mathbf{n}}$ through equation (32).

Step 2.5 Increment n.

Step 3. Return the value functions $\bar{V}$ and coefficients θ_*.

Figure 8. Stochastic Gradient Search by Approximate Value Iteration.

Note that in step 2.2, S^M is actually a simulation model which samples the state transition function and returns a sample realization ω for the immediate cost $C_n(S_n, \boldsymbol{a}_n)$, during the decision period τ, and the status of successor state S_{n+1}.

VI.C Artificial Neural Network with Temporal Difference Learning

VI.C.1 Artificial Neural Network Model

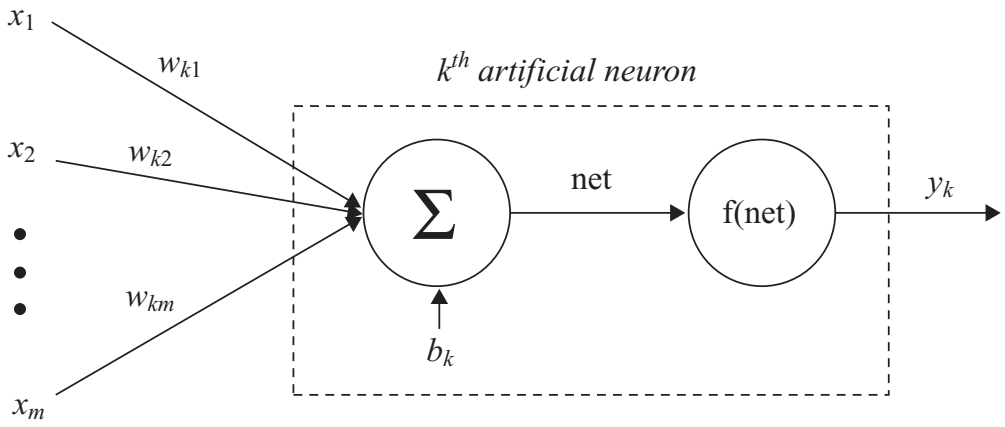

Figure 9. A Basic Artificial Neural Network Model (Kantardzic, 2011)

Figure 9 illustrate how does a basic artificial neural network model works. In this model, there are m inputs $x_1, ..., x_m$ and y_k is the output which one wishes to estimate. w_{ki} is a weight associated with input x_i. b_k is a constant term which is included in the model.

In this artificial neural network model, a sample realization corresponding to the input vector $X(t) = (x_1(t), ..., x_m(t))$ at iteration t is already known and denoted by $f_k(t)$. After the input vector X go through the artificial neuron k, an output value of $y_k(t)$ will be produced. An error $e_k(t) = f_k(t) - y_k(t)$ will appear. To perform the learning process of the artificial neural network model, one need to adjust the value of each input weight w_{ki}, then ultimately find the vector of $W = (w_{k1}, ..., w_{km})$ that solves equation (37) during the current iteration:

$$\min_{W} \mathbb{E}\frac{1}{2}(f_k(t) - y_k(t))^2 \tag{37}$$

The model in figure 9 only represents a basic artificial neural network model with single neuron. The actual architecture of artificial neural network model would be much more complex as figure 10 shows.

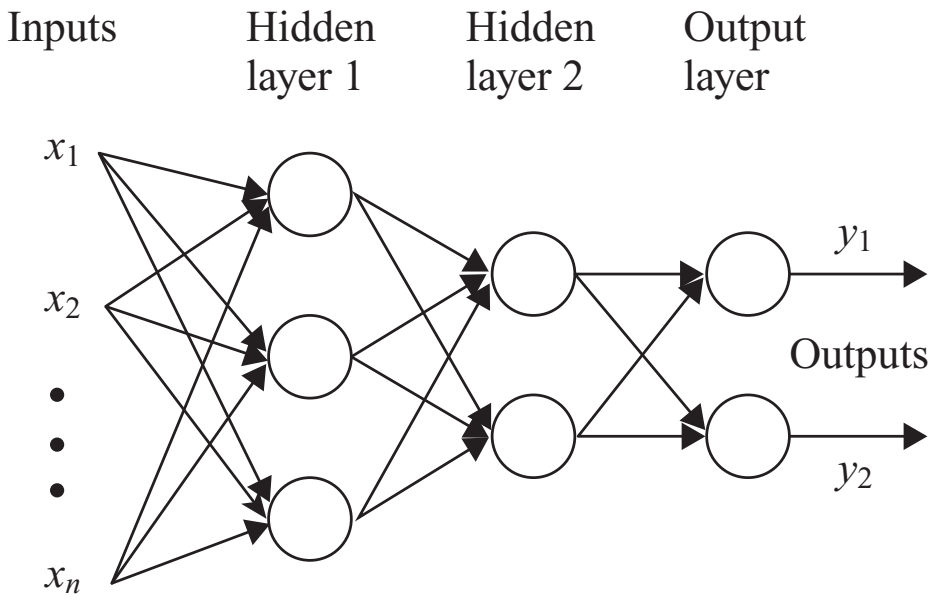

Figure 10. A Complex Architecture of an Artificial Neural Network (Kantardzic, 2011)

The artificial neural network model shown in figure 10 is a multi-layer (MLPs: multi-layer perceptrons) and multi-output model. For each layer, there is more than one neuron. As the figure shows, neurons between two layers are interacting with each other through the weight w_{ij}. This feature makes artificial neural network is capable to capture both linear and non-linear relationship from the input and

output, without any presumption about the parametric distribution from the observation. Also, ANN model can be driven and self-adaptive from the the input.

VI.C.2 ANN Approximation Scheme

Controlling Mechanism

Artificial neural networks (ANN) are suitable to address machine learning and pattern recognition problems (Kantardzic, 2011). In approximate dynamic programming, an ANN is able to perform as an effective tool to approximate the state-dependent value (e.g., $C_\tau(\boldsymbol{S}, \boldsymbol{a})$ in section V.B.3) from a large amount of input iteratively. In our research, the ANN not only contributes its feature for learning the state values for the system, but it also provides control information for the production operation.

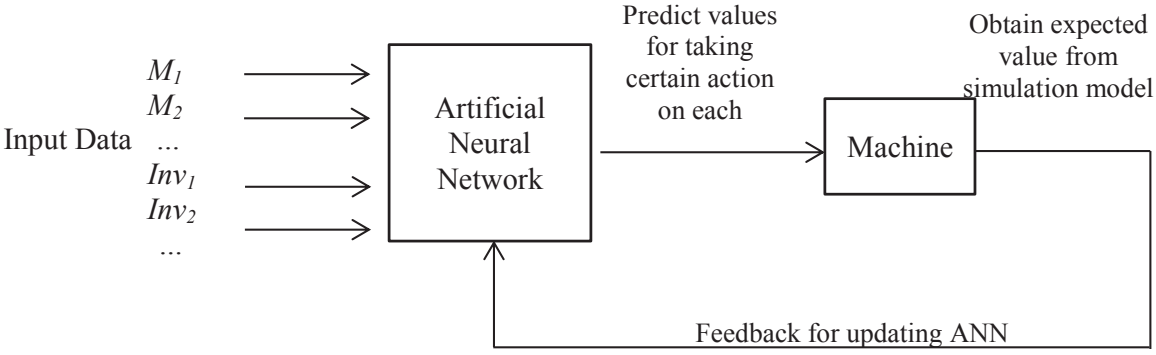

Figure 11. Mechanism of ANN Model.

As figure 11 shows, at each decision point, we have input data which consists of both the machine status vector M and the inventory level vector Inv: $S = (M, Inv)$. After the input of this data is processed by the ANN model, the

response vector $Y(y_{1_k}, ..., y_{f_k})$ is obtained to estimate the average rewards of taking different actions on machine k. For instance, response y_{f_k} reflects the estimated average reward of taking action f on machine k in the next decision period. The system will be controlled by choosing the optimal action $a_k^* = \arg\min_{f_k \in F_k} y_{f_k}$ which minimizes the estimated reward y_{f_k} to $y_{f_k}^*$, where $y_{f_k}^* = \min(Y(y_{1_k}, ..., y_{f_k}))$. (Note that F_k is the set of actions can be performed on machine k) After the best action a_k^* is chosen, the simulation model will continue executing, and the expected average reward $f_\tau(S, a_k^*)$, associated with taking action a_k^* at state S on machine k for decision period τ, will be returned from the simulation model as in equation (38), and compared with the estimated value $y_{f_k}^*$ to provide adjusting information of the ANN model.

$$f_\tau(S, a_k^*) = \frac{C_\tau(S, a)}{\tau} \tag{38}$$

ANN Architecture

The architecture of an ANN is defined by the characteristics of a node and the characteristics of the node's connectivity in the network. In practical applications, the most widely used architecture of the ANN model is a multilayer feedforward network (Kantardzic, 2011). A general multilayer network model is tested to control the system and approximate the state-action pair value. The multilayer network model updates via iterations as described in figure 11. However, since there is no boundary in a general multilayer network model, starting with an initial unlearned

network will easily lead the system to be in states of poor quality (e.g., the state which has item A inventory level extremely high, but item B extremely low). If the network updates on these poor states without giving information for continuous improvement, it will fail to control and optimize the system. To prevent the system from stalling in states of poor quality, a characterised multi-layer ANN model which adds the feature of an (s, S) policy is proposed as figure 12 shows:

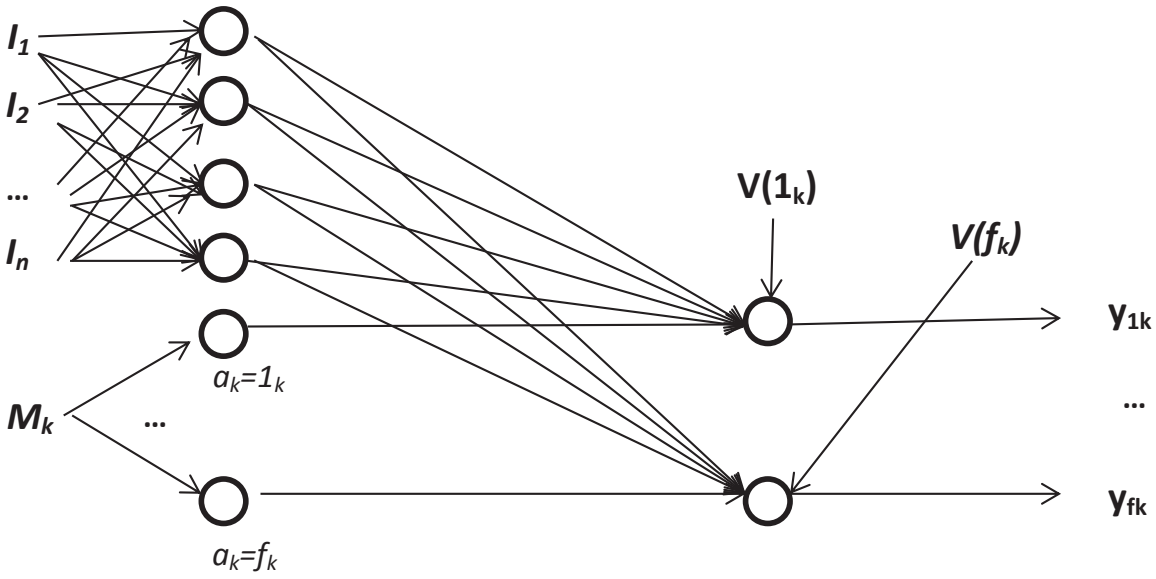

Figure 12. ANN Architecture.

In the multilayer ANN model, the inputs are the inventory levels for each of the respective items and the machine status M_k, while the outputs are the $y_{1_k}, ..., y_{f_k}$ to estimate the average rewards of choosing action f on machine k. The inputs of the inventories $Inv_1, Inv_2, ..., Inv_n$ follow the general multilayer perceptrons (MLPs) architecture. The inputs of machine status M_k will be compared with the neurons of possible actions $a_k = (1_k, ..., f_k)$ on machine k. If the machine status M_k is not

equal to action on the neuron, the output value on the neuron is 1, otherwise, the output value on the neuron is 0. For example, if $M_k = 1$, the output value on '$a_k = 1$' is 0, and the output values on other neurons associated with M_k are 1.

At the neuron associated with the output value of y_{f_k}, there is a penalty value $V(f_k)$ added on this neuron to prevent the system from being in states with poor quality. The penalty value $V(f_k)$ is generated via a (s, S) policy which can be simply obtained by a heuristic search, as defined in equation (39). Note that f_k is not only a value of action, but also the index of product which is decided to produce (or perform changeover) by the corresponding action.

$$V(f_k) = \begin{cases} M, & \text{if } I_{f_k} > S_{f_k} \\ 0, & \text{if } s_{f_k} \leq I_{f_k} \leq S_{f_k} \\ -M, & \text{if } I_{f_k} < s_{f_k} \end{cases} \tag{39}$$

In equation (39), s_{f_k} and S_{f_k} represent respective values for the parameters s and S in the (s, S) inventory policy for product f_k, while M is a large positive number. In this ANN model, $V(f_k)$ acts as a boundary to ensure that the inventory level of product f_k does not become extremely high or low. $V(f_k)$ can keep the production system in manageable states, and also ensure that the parameters in the ANN model are not updated by using information with poor states continuously.

Updating Procedure

In the artificial neural network model, there is a weight w_{ij} associated with input (or neuron) i and neuron j. To minimize the error ($e_i(t) = f_i(t) - y_i(t)$) between the predicted value $y_i(t)$ from the ANN model and the value $f_i(t)$ that would be computed from the simulation model by following a sample path, the weight vector of $W = (w_{11}, ..., w_{ij})$ needs to be updated iteratively, thus improve the predict accuracy. Therefore, we use the backpropagation method as the updating procedure for the weight w_{ij}, as figure 13 shows:

Step 1. Initialization. Set all $w_{ji}(0) = 0$.

Step 2. For iteration $n = 1, 2, ...,$ do:

Step 2.1. Compute net value of j^{th} neuron:

$$v_j(n) = \sum_{i=1}^{m} w_{ji}(n) x_i(n)$$

where m is the number of inputs for j^{th} neuron

Step 2.2. Compute the *local gradient*:

$$\delta_j(n) = \begin{cases} e_j(n)\varphi'(v_j(n)), & \text{for output layer} \\ \varphi'(v_j(n)) \sum_{k \in D} \delta_k(n) w_{kj}(n), & \text{for hidden layer} \end{cases}$$

Step 2.3. Update weight $w_{ji}(n)$.

$$\Delta w_{ji}(n) = \eta \delta_j(n) x_i(n) + \alpha \Delta w_{ji}(n-1) \tag{40}$$

Figure 13. ANN Updating Procedure.

Note that $y_i(n) = \varphi(v_j(n))$ is the output value and $\varphi(v_j(n))$ is an activation function (e.g., Sigmoid function). D denotes the set of all nodes on the next layer that are connected to the node j. In equation (40), where α is usually a positive number called *momentum constant* and η is the learning-rate parameter. $\Delta w_{ji}(n-1)$ is the correction of the weight factor for a previous $(n-1)^{st}$ sample. The addition of the momentum term smoothes the weight being updated and tends to resist erratic weight changes resulting from gradient noise or high-spatial frequencies in the error surface. (Kantardzic, 2011)

VI.C.3 Temporal Difference Learning

Temporal difference (TD) learning is a supervised learning algorithm which is often used in reinforcement learning to predict a measure of the total amount of reward expected about the future. TD learning is viewed as a combination of Monte Carlo method and dynamic programming by scholars (Sutton and Barto, 1998). The Monte Carlo method is used in TD learning to create the sample path based on the policy chosen. And TD learning is related to dynamic programming because it follows a bootstrapping process which estimate the current value based on the previously learned estimate. At each learning step, a prediction is made from the knowledge of previous learning steps, after the observation for this step is available, a new prediction will be updated based on the difference of the previous prediction and the observation. Over adequate successive steps of learning, the prediction will be adjusted to better match the observation, thus become more accurate.

The mathematical formulation of temporal difference in approximate dynamic programming is shown as equation (41):

$$\delta_\tau^\pi = C(S_\tau^n, a_\tau^n, W_{\tau+1}^n) + \bar{V}_{\tau+1}^{n-1}(S_{\tau+1}^n) - \bar{V}_\tau^{n-1}(S_\tau^n) \tag{41}$$

In equation (41), the component of $C(S_\tau^n, a_\tau^n, W_{\tau+1}^n) + \bar{V}_{\tau+1}^{n-1}(S_{\tau+1}^n)$ is the sampled observation of being in state S_τ, while $\bar{V}_\tau^{n-1}(S_\tau^n)$ is the current estimate of the value of being in state S_τ.

To update the prediction of the value $\bar{V}_t^n(S_t)$ of being in in state S_t, the previous temporal differences should be cumulated as equation (42) shows:

$$\bar{V}_t^n(S_t) = \bar{V}_t^{n-1}(S_t) + \sum_{\tau=t}^{T} \delta_\tau^\pi \tag{42}$$

One may think that those later estimates of the differences should be given more weight than the later ones, thereby an artificial discount factor λ can be introduced into equation (42) as a result. Meanwhile, a time discount factor γ can be also introduced into equation (42) to capture the time effect in the model. Then, equation (42) will come to be:

$$\bar{V}_t^n(S_t) = \bar{V}_t^{n-1}(S_t) + \sum_{\tau=t}^{T} (\gamma\lambda)^{\tau-t} \delta_\tau^\pi \tag{43}$$

VI.C.4 ADP Algorithm: Artificial Neural Network by Temporal-Difference Learning

By using the artificial neural network model we proposed in section VI.C.2 to approximate the average cost for the system during each decision period, and Temporal-Difference Learning method we introduced in section VI.C.3, we developed a ADP algorithm by using the backpropagation to update the weight w_{ij} in the ANN model, as figure 14 shows:

Step 0. Initialization: Set all $w_{ji}(0) = 0$ in ANN model.

Step 1. Generate a sample path.

Step 2. Do for $n = 1, 2, ...$

Step 2.1. Find action a_k^* to minimise the output value in ANN model y_{f_k}:

$$a_k^* = \arg \min_{f_k \in F_k} y_{f_k}.$$

Step 2.2. Obtain information for decision period τ through the simulation model:

$$(C_n, \tau_n, S_{n+1}) \leftarrow S^M(S_n, a_k^*)$$

Step 2.3. Update weight $w_{ji}(n)$ by the procedure as described in section VI.C.2.

Step 2.4 Increment n.

Step 3. Return weight $w_{ji}*$ in the Artificial Neural Network Model.

Figure 14. Artificial Neural Network by TD Learning Algorithm.

CHAPTER VII

NUMERICAL EXAMPLE

VII.A Example Description

In chapter VII, the model formulated in chapter V is implemented to a simple numerical example from the problem which is discussed in section II.B. The solution methods 1, 2 and 3 proposed in chapter VI is performed on the numerical example.

Consider a two-machine and make-to-stock production system as figure 15 shows. Machine 1 produces two types of intermediate components IC_1 and IC_2, whereas Machine 2 produces two types of finished products FP_1 and FP_2 by consuming the intermediate components IC_1 and IC_2.

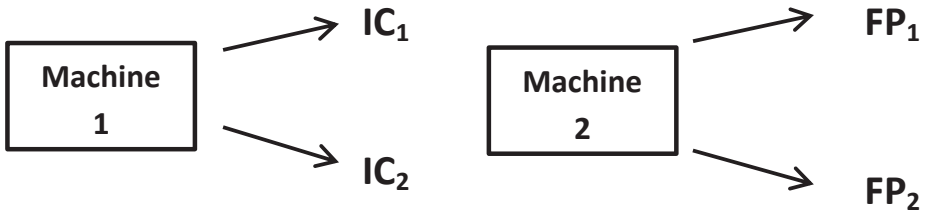

Figure 15. 2-2-2 example.

Significant setup times occur during changeovers at both of the machines. The objective of the problem is to find an effective control policy to operate the

production system. This problem can be considered as an $I = 2$, $J = 2$, $K = 2$ $(2 - 2 - 2)$ example for the model formulated in chapter V.

The experimental parameters set up for this numerical example can be found in tables 5, 6 and 7.

TABLE 5

Parameters for Finished Product.

Product i	T_i	h_i	b_i	λ_i	q_i
1	0.15	1	5	5	0.5
2	0.20	1	8	10	0.4

TABLE 6

Parameters for Intermediate Product.

Product j	T_j	h_j
1	0.10	2
2	0.15	2

TABLE 7

Parameters for Machine.

Machine k	Pr_k
1	40
2	30

VII.B Results

In chapter VI, we proposed three methods which can be used to solve the problem. Although the problem itself suffers from the "curse of dimensionality", the formal Dynamic Programming method is still applicable to the simple numerical example in chapter VII. We have also used Dynamic Programming to solve the numerical example, then compare the results with the methods we proposed in chapter VI. In the Dynamic Programming method, we determined the arrival of the demand and the demand arrival size by using the estimated approximation of the compound Poisson processes in section V.B.2.

Table 8 shows the measurements of the performance for the adjusted (s, S) policy, the Linear Models with Basis Function, the ADP-ANN approach, and a policy generated from a formal dynamic programming (DP) approach by using approximation for the uncertainties in this numerical example. Since all the uncertainties were dealt with approximation in the formal DP approach and this example is relatively small, it required very little computational effort to obtain the optimal policy for the formal DP approach compared to the other two methods which were updating and searching a better solution in conjunction with the simulation model. However, the result from the formal DP approach has the worst performance of the three methods. The primary reason is that its average cost is much higher than that of the other two approaches. It has a relatively high level (21.38%) of unmet demand as well, resulting in a large backorder cost. Also, note

that the number of changeovers required in the formal dynamic programming approach is much higher than in the other approaches.

TABLE 8

Measures of (s, S) and ANN for 2-2-2 Example.

Main Measures	**Formal DP**	**Linear Model**	(s, S)	**ADP-ANN**	**ADP-ANN v.s DP**	**ADP-ANN v.s Linear Model**	**ADP-ANN v.s** (s, S)
Average Cost	62.702	55.8196	55.368	51.247	22%	9%	8%
Holding Cost	17,431	47,836	49,533	43,959	(-)60%	9%	13%
Backorder Cost	41,307	1742	75.842	225.25	18238%	673%	(-)66%
Percentage of Unmet Demand	21.38%	12.58%	4.43%	9.12%	135%	38%	(-)51%
Average Inventory (Intermediate)	0.676	18.2573	19.1077	17.9128	-96%	2%	7%
Average Inventory (End)	(-)16.116	15.2764	15.2392	13.1379	223%	16%	16%
Number of Changeover (Intermediate)	300	182	174	173	73%	5%	1%
Number of Changeover (End)	300	264	179	239	26%	14%	(-)25%
Lack of Intermediate Product	N/A	326	249	97	(-)100%	236%	141%

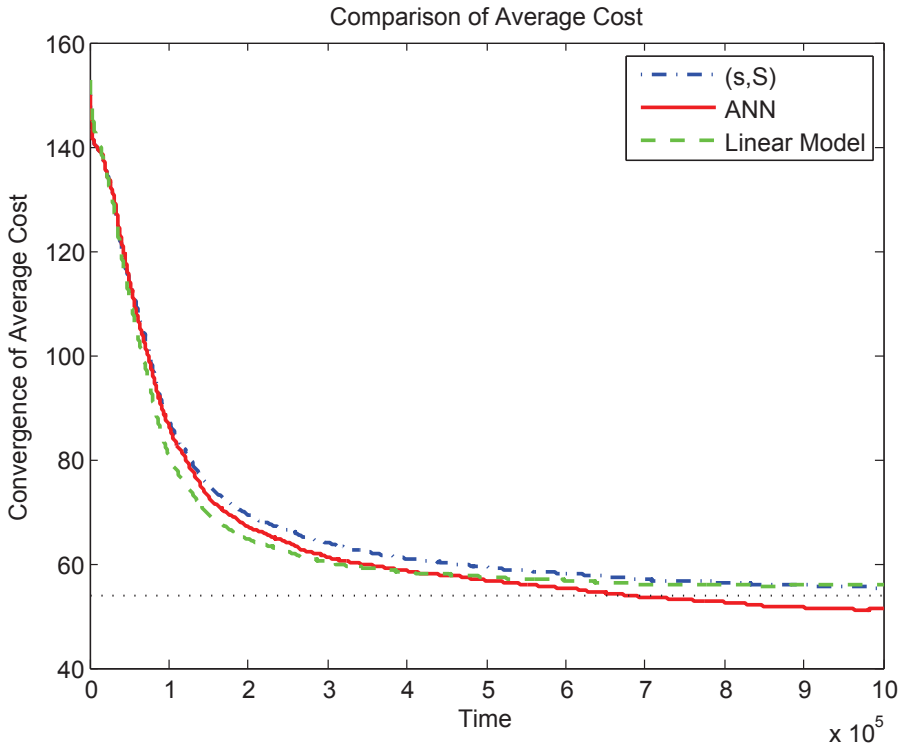

Figure 16. Comparing the Convergences of Average Cost for 2-2-2 Example.

In this section, we also compared the results from the ADP algorithm in section VI.C using the ANN model and the results from the ADP algorithm approximated by Linear Regression Models with Basis Function in section VI.B, to the results from a adjusted (s, S) policy in section VI.A obtained by a heuristic search.

The linear regression model with basis function reduced the average cost much more than formal DP approach, since it reduced the holding cost, and the number of changeover for both intermediate and end products much. However, it does not show much better performance than the adjusted (s, S) policy. Although the holding cost is lower for the linear regression approximation, the backorder cost

is much higher than the adjusted (s, S) policy. Also, the unmet demand for the linear regression approximation is much higher. Some other important performance measurements are the number of changeover for both of intermediate and end product, as well as the lack of intermediate product. The lack of intermediate product counts the time that there is not enough intermediate products when it is required to produce a certain type of end product. This awkward situation is which we want to avoid during the production process, since it would waste a lot of time and resource. Compare to the adjusted (s, S) policy, the linear model definitely have worse performance on these measurements.

The ADP-ANN approach results in a little slightly higher backorder cost but a much lower inventory level than the adjusted (s, S) policy and the average cost is much lower (8%) by using the ADP-ANN approach. Compared to the adjusted (s, S) policy, the numbers of changeovers at the intermediate machine are almost the same for the two methods, and the number of changeovers for the end product machine is larger for the ADP-ANN approach than for the adjusted (s, S) policy. However, the lack of intermediate product is much lower by using the ADP-ANN approach than the adjusted (s, S) policy, which indicate that the ADP-ANN method can control the system very smoothly. The ADP-ANN method shows a better performance on the numerical example than the adjusted (s, S) policy.

Figure 16 shows the comparison of convergence curves for the average cost over time from 0 to 10^6 using the ADP-ANN approach, the Linear Regression Model with Basis Function approximation and the adjusted (s, S) policy to control the

system. The X-axis is the number of the time units for the simulation period, while the Y-axis represents the average cost. Note that the final average cost for the ADP-ANN approach is close to 50, the final average cost for the linear model approximation is close to 56, and the final average cost for adjusted (s, S) policy is close to 55.

The linear model approximation shows a faster convergence than the ADP-ANN approach and the adjusted (s, S) policy. However, it does not give us a comparable reduction for the final average cost.

Figure 17 shows the comparison of the learning difference by applying linear regression approximation and ANN-ADP approach to approximate the state values in the numerical example. Compare to the ANN-ADP approach, the learning difference is smaller before $t = 300000$ by using the linear regression approximation with basis function. However, for a long term run of the simulation model, the linear regression approximation does not reduced the learning difference as much as the ADP-ANN approach

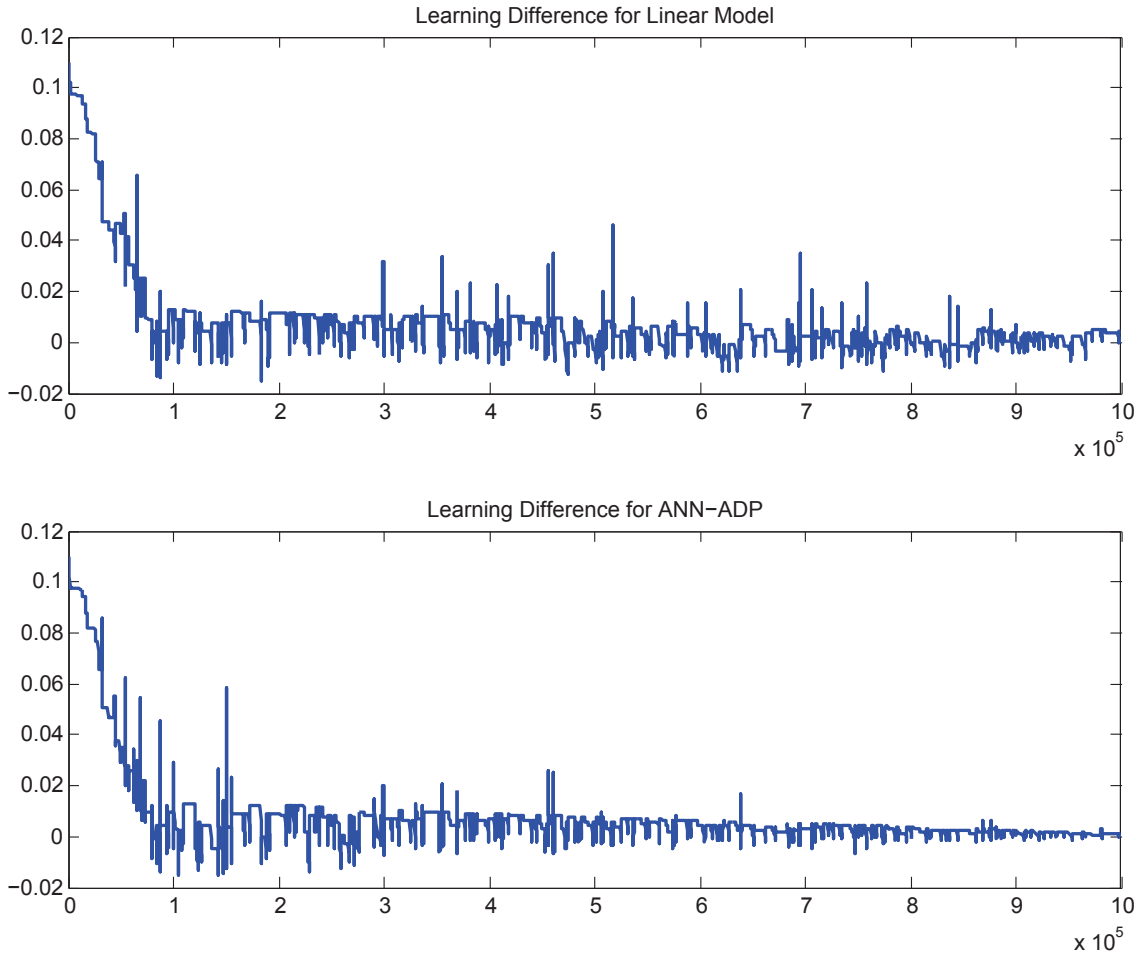

Figure 17. Learning Difference for Linear Model and ADP-ANN.

In summary, all the three methods proposed in chapter VI works better than the formal DP methods on the numerical example. The ADP-ANN approach perform best among the three methods, and the adjusted (s, S) policy shows slightly better results than the linear regression approximation.

CHAPTER VIII

GE PROBLEM SOLUTION

VIII.A Problem Description

By examining the results of the numerical study in chapter VII, approximate dynamic programming could be applied to optimize the control process for the complex GE production system described in section II.B for which the formal Dynamic Programming is not directly applicable.

However, although the linear regression model with basis function described in section VI.B can avoid the "curse of dimensionality" for the extreme large state space in the complex production system (GE), the number of the linear regression models will grow very fast with the increase of the number of machines, which means the action space A still suffer from the "curse of dimensionality". Therefore, only Methods 1 and 3 which we proposed are able to be generalized to solve the control optimization problems for the complex system, such as the GE system.

As it is described in section II.B, the GE production system has five work centers, each work center is associated with one to four different products. There are total nine intermediate components and thirteen end products. The action space A is $4 * 5 * 3 * 2 * 4 = 480$, which is already a very large space, not to mention the

state space is added into the problem.

In chapter VIII, we are trying to apply the adjusted (s, S) control policy to solve the GE problem. After that, the results we obtain from the adjusted (s, S) control policy can be used as parameters for the artificial neural network model we proposed in section VI.C to approximate the average cost of the production system, and thus optimize the control process through the approximate dynamic programming algorithm in figure 14.

The parameters for GE problem can be found in table 9, 10 and 11 as following:

TABLE 9

Parameters for Finished Product.

Product i	T_i	h_i	b_i	λ_i	q_i
1	150	1	8	188	0.13
2	150	1	6	169	0.29
3	150	1	6	174	0.27
4	150	1	6	180	0.19
5	150	1	4	58	0.89
6	150	1	8	188	0.13
7	150	1	6	169	0.29
8	150	1	6	174	0.27
9	150	1	5	108	0.85
10	150	1	4	14	0.81
11	150	1	5	129	0.66
12	150	1	6	171	0.28
13	150	1	5	41	0.86

TABLE 10

Parameters for Intermediate Product.

Product j	T_j	h_j
1	150	2
2	150	2
3	150	2
4	150	2
5	150	2
6	150	2
7	150	2
8	150	2
9	150	2

TABLE 11

Parameters for Machine.

Machine k	Pr_k	Machine Name
1	1	FL
2	1	FU1
3	1	FU2
4	0.5	Type 1 Coating
5	0.5	Type 2 Coating

VIII.B Results

In this section, the performance of the ANN-embedded ADP algorithm is measured in comparison with a adjusted (s, S) policy. To obtain the optimal control variable s and S in the adjusted (s, S) policy, we performed a tabu search on the

simulation model. If the inventory level is less than 100, the increment (decrement) for the tabu search is 10, otherwise, the increment (decrement) is 100. Table 12 shows the results of s and S from the optimization via simulation by using the adjusted (s, S) policy.

TABLE 12

Optimal (s, S) policy.

Coated Rack	s	S	Fabricated Rack	s	S
1	700	1300	A	500	1000
2	400	700	B	400	900
3	100	200	BXL	700	1200
4	600	1000	A1	800	1300
5	10	20	B1	300	600
6	500	800	B2	200	400
7	200	400	B3	40	80
8	400	600	C2	60	100
9	60	100	C4	200	400
10	20	40			
11	120	200			
12	300	600			
13	200	400			

Figure 18 shows the comparison of convergences for the average cost by using the ANN model and the adjusted (s, S) policy to control the system for the GE problem. The adjusted (s, S) policy converges slightly faster than the ADP-ANN method, however, the final average cost it reached is much higher than the ADP-ANN method.

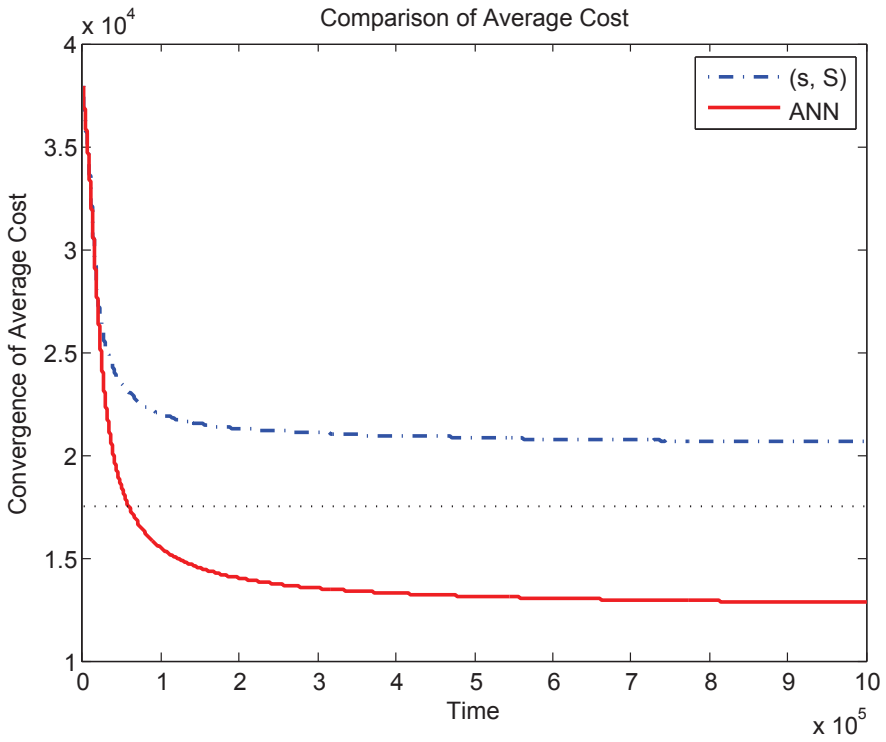

Figure 18. Comparing the Convergences of Average Cost for (s, S) Policy and ADP-ANN.

Table 13 compares the performance measures of the two control methods. The average cost for the ANN model converges to 12,834 while the average cost for the adjusted (s, S) policy converges to 20,672. By using ANN model as the second level policy to control the system, the average cost is reduced significantly, by 37.92%, since both of the inventory levels for intermediate and end products are significantly reduced. The percentages of unmet demand for the end products are 0 for both of the two methods. The number of changeovers is significantly increased by using the ANN model to control the system than by using the adjusted (s, S)

policy; through the use of the ANN model the production system can be operated more efficiently, thus keeping the inventories at lower levels. Also, note the lack of intermediate product is significantly reduced, which means the two production stages (for intermediate product and end product) are interacting with each other very well. The undesired situation that in short of the associated intermediate product when the production of end products is required is rarely happening by using the ADP-ANN to control the production system.

TABLE 13

Measures of (s, S) and ANN for Real Case.

Main Measures	(s, S)	**ADP-ANN**	**Reduction**
Average Cost	20,672	12,834	38%
Holding Cost	$2.0671 \text{x} 10^{10}$	$1.2825 \text{x} 10^{10}$	61%
Backorder Cost	0	0	N/A
Percentage of Unmet Demand	0%	0%	N/A
Average Inventory (Intermediate)	3,358	1,841	45%
Average Inventory (End)	3,524	2,002	43%
Number of Changeover (Intermediate1)	422	3,569	N/A
Number of Changeover (Intermediate2)	558	4,356	N/A
Number of Changeover (Intermediate3)	246	5,851	N/A
Number of Changeover (End1)	453	4,122	N/A
Number of Changeover (End2)	0	0	N/A
Lack of Intermediate Product	6.74%	0.86%	87%

Figure 19 shows the change of the learning difference of ADP-ANN method performed on the GE problem. Although the ANN model started with a very large learning difference, after a number of iterations, the learning difference converges to a small value.

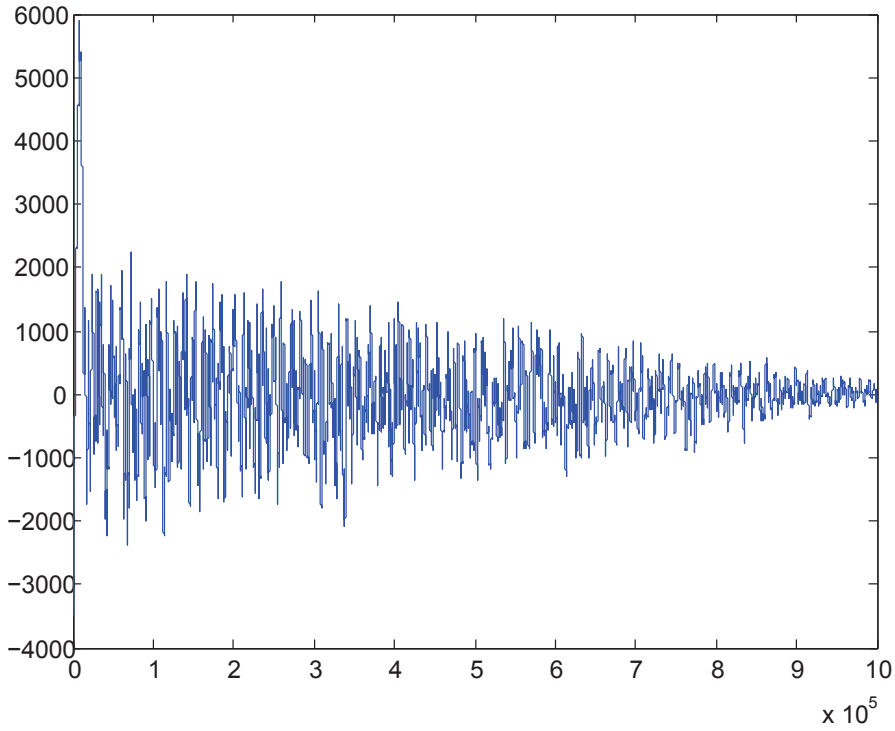

Figure 19. Learning Difference for Linear Model and ADP-ANN.

CHAPTER IX

SUMMARY AND FUTURE RESEARCH

IX.A Summary

In order to quickly respond to various requirements from the market, the flexibility is considered as one of essential features in the design of a production system. However, it will make a production system more dynamic, thus many operational difficulties will arise. The production control policy is often driven by the inventories and developed via analytical models. With the increase of uncertainties, varieties of products and system dynamics, the normal analytical models often fail to obtain a optimal or near optimal control policy for the production system.

In our research, to look for a better control policy of a complex dynamic production system with multiple working stations, intermediate components, uncertainties and significant changeover time, three methods are developed in chapter VI:

- Optimization via simulation by using an adjusted (s, S) inventory control

policy.

- An Approximate Dynamic Programming method by using Linear Regression Models with Basis Functions to approximate the value function of the production system.

- An Approximate Dynamic Programming method by using an Artificial Neural Network model to approximate the value function of the production system. The structure of the Artificial Neural Network model is improved in order to capture the characteristics of the system and also include the information from a adjusted (s, S) policy.

The methods we proposed in chapter VI are first tested on a small numerical example in chapter VII. For the small multiple machines with intermediate products problem, approximate dynamic programming methods shows much better performance than using formal dynamic programming method with approximation of the uncertainties in the production system. However, compared to the adjusted (s, S) policy, the advantage of Approximate Dynamic Programming would depend on the schemes to approximate the value function.

In the small numerical example, by using linear regression model with basis functions as the approximation scheme, the results is not improved compared to the adjusted (s, S) policy, although it converges faster. However, by using ADP-ANN approach as the second level policy to control the system, the average cost is

reduced significantly by 8%, but the percentage of unmet demand is higher by using the ADP-ANN model. Since the objective is to reduce the cost and the shortage of final products, is taken into account as backorder cost in the objective function, the ANN-ADP approach still performs better than the adjusted (s, S) policy. It is noteworthy that by using the ANN-ADP approach, the lack of intermediate products is much less of a problem, this indicates that the ANN-ADP approach is capable of controlling the system more intelligently.

The adjusted (s, S) policy and the ADP-ANN algorithm are also applied to solve a complex problem from GE wire rack production system which suffers from the "curse of dimensionality" in chapter VIII. As compared to the small numerical example, the ADP-ANN approach is even much more pronounced for such more complex system than the adjusted (s, S) policy.

From our research, Approximate Dynamic Programming is found to be an effective way to predict such kind of complex production systems and improve the operating process as well as their productivity. Moreover, the proposed methods can be adjusted and applied to solve other stochastic-dynamic decision problems from other fields, such as supply chain, revenue management, and finance.

IX.B Future Research

Future research directions stemming from current work in this dissertation are as follows:

- There are many statistical learning models that can be used as the approximation scheme for the approximate dynamic programming method, such as support vector regression, random forest and kernel regression. One of them may capture the characteristics of such kind of problems better, thus improving the control policy for the complex production system.

- Since the dynamic state-dependent policy generated via the approximate dynamic programming algorithm is predicated on the current information gathered from the system, it does not correspond to an easily implemented policy like the adjusted (s, S) policy. To handle this problem, some classification techniques, such as decision tree and k-nearest neighbors algorithm, can be used to summarize the generated control policy and make it more implementable for operators of a complex production system.

- This research can be also extend to the overall GE production system, and is not limited to the wire rack production system only. The difficulty for the extension is to find a simulation software which can model the system efficiently and embed the complex algorithm easily, rather than to code everything through the programming language inefficiently.

REFERENCES

T. Altiok and G.A. Shiue. Single-stage, multi-product production/inventory systems with backorders. *IIE Transactions*, 26(2):52C61, 1994.

T. Altiok and G.A. Shiue. Single-stage, multi-product production/inventory systems with lost sales. *Naval Research Logistics*, 42:889C913, 1995.

T. Altiok and G.A. Shiue. Pull-type manufacturing systems with multiple product types. *IIE Transactions*, 32(2):115C124, 2000a.

T. Altiok and G.A. Shiue. Pull-type manufacturing systems with multiple product types. *IIE Transactions*, 32(2):115–124, 2000b.

R.G. Askin and J.B. Goldberg. *Design and Analysis of Lean Production System*. John Wiley & Sons, Inc., New York, NY, ltd edition, 2002.

R. Bellman. and S. Dreyfus. The optimizing-simulator: Merging simulation and optimization using approximate dynamic programming. *Mathematical Tables and Other Aids to Computation*, 13:247C251, 1959.

O. Benedettini and B. Tjahjono. Towards an improved tool to facilitate simulation modelling of complex manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 43:191–199, 2009.

D. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, vol.2, 3rd ed edition, 2007.

D. Bertsekas and J. Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, Belmont, MA, 1996.

K.E. Bourland. Production planning and control links and the stochastic economic lot scheduling problem. *Management Science*, Working paper no. 299:The Amos Tuck School of Business Administration, Hanover, 1994.

K.E. Bourland and C.A. Yano. The strategic use of capacity slack in the economic lot scheduling problem with random demand. *Management Science*, 40(12): 1690C1704, 1994.

P. Brander and R. Forsberg. Determination of safety stocks for cyclic schedules with stochastic demands. *International Journal of Production Economics*, 104: 271C295, 2006.

P. Brander, E. Lven, and A. Segerstedt. Lot sizes in a capacity constrained facility c a simulation study of stationary stochastic demand. *International Journal of Production Economics*, page 375C386, 2005.

J. Bruin. Cyclic multi-item production systems. *Proceedings of Analysis of Manufacturing Systems*, page 99C104, 2007.

D.D. Eisenstein. Recovering cyclic schedules using dynamic produce-up-to policies. *Operations Research*, 53(4):675C688, 2005.

N. Erkip, R. Gll, and A. Kocabiyikoglu. A quasi-birth-and-death model to evaluate fixed cycle time policies for stochastic multi-item production/inventory problem. *Proceedings of MSOM conference, Ann Arbor*, 2000.

A. Federgruen and Z. Katalan. The stochastic economic lot scheduling problem: Cyclical base-stock policies with idle times. *Management Science*, 42(6):271C295, 1996a.

A. Federgruen and Z. Katalan. Determining production schedules under base-stock policies in single facility multi-item production systems. *Management Science*, 42(6):271C295, 1996b.

A. Federgruen and Z. Katalan. Customer waiting-time distributions under basestock policies in single-facility multi-item production systems. *Operations Research*, 46(6):883C898, 1998.

J.C. Fransoo. Demand management and production control in process industries. *International Journal of Operations and Production Management*, 12(7/8): 187C196, 1992.

G. Gallego. Scheduling the production of several items with random demands in a single facility. *Management Science*, 36(12):1579C1592, 1990.

G. Gallego. When is a base stock policy optimal in recovering disrupted cyclic schedules? *Naval Research Logistics*, 41:317C333, 1994.

A. Gascon, R.C. Leachman, and P. Lefrancois. Multi-item, single-machine scheduling problem with stochastic demands: a comparison of heuristics. *International Journal of Operations Research*, 32:583C596, 1994.

G.A. Godfrey and W.B. Powell. An adaptive, distribution-free approximation for the newsvendor problem with censored demands, with applications to inventory and distribution problems. *Management Science*, 47(8):1101C1112, 2001.

A. Gosavi. Reinforcement learning: A tutorial survey and recent advances. *IIE Transactions*, 21(2):178C192, 2009.

S.E. Grasman, T.L. Olsen, and J.R. Birge. Setting basestock levels in multiproduct systems with setups and random yield. *IIE Transactions*, 40(12):1158C1170, 2008.

Y. Han and C. Zhou. Dynamic sequencing of jobs on conveyor systems for minimizing changeovers. *The International Journal of Advanced Manufacturing Technology*, 49:1251–1259, 2010.

R.A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. Press, 1960.

W.L. Hsu. On the general feasibility test of scheduling lot sizes for several products on one machine. *Management Science*, 29:93C105, 1983.

J. Hu, S. Nananukul, and W. Gong. A new approach to (s,s) inventory systems. *Journal of Applied Probability*, 30(4):898–912, 1993.

M. Huang, W.H. Ip, K.L. Yung, X. Wang, and D. Wang. Simulation study using system dynamics for a conwip-controlled lamp supply chain. *The International Journal of Advanced Manufacturing Technology*, 32:184–193, 2007.

T. Jaakkola, M.I. Jordan, and S.P. Singh. Convergence of stochastic iterative dynamic programming algorithms. *In Advances in Neural Information Process-ing Systems*, 6:703C710, 1994.

M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons, Inc., Hoboken, NJ, 2nd edition, 2011.

U.S. Karmarkar and J. Yoo. Stochastic dynamic product cycling problem. *European Journal of Operational Research*, 73:360C370, 1994.

G.N. Krieg and H. Kuhn. A decomposition method for multi-product kanban systems with setup times and lost sales. *IIE Transactions*, 34(7):613C625, 2002.

G.N. Krieg and H. Kuhn. Analysis of multi-product kanban systems with statedependent setups and lost sales. *Annals of Operations Research*, (125): 141C166, 2004.

S. Kumar and R. Sridharan. Simulation modeling and analysis of tool sharing and part scheduling decisions in single-stage multimachine flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 23:361–370, 2007.

R.C. Leachman and A. Gascon. A heuristic policy for multi-item, single-machine

production systems with time-varying stochastic demands. *Management Science*, 34(3):377C390, 1988.

R.C. Leachman, Z.K. Xiong, A. Gascon, and K. Park. An improvement to the dynamic cycle lengths heuristic for scheduling the multi-item, single-machine. *Management Science*, 37(9):1201C1205, 1991.

H.-S. Lee and M.M. Srinivasan. *The (s,S) policy for the production/inventory system with compound Poisson demands*. University of Michigan, Ann Arbor, MI, April 1988.

N. Lhndorf and S. Minner. Simulation optimization for the stochastic economic lot scheduling problem. *IIE Transactions*, 45(7):796–810, 2012.

C. Lin, T.S. Baines, J. O'Kane, and D. Link. A generic methodology that aid the application of system dynamics to manufacturing system modelling. *International Conference on Simualtion*, (457):344–349, 1998.

D.M. Markowitz and L.M. Wein. Heavy traffic analysis of dynamic cyclic policies: A unified treatment of the single machine scheduling problem. *Operational Research*, 49(2):246C270, 2001.

D.M. Markowitz, M.I. Reiman, and L.M. Wein. The stochastic economic lot scheduling problem: Heavy traffic analysis of dynamic cyclic policies. *Operational Research*, 48(1):136C154, 2000.

M. Masin and V. Prabhu. Awip: A simulation-based feedback control algorithm for scalable design of self-regulating production control systems. *IIE Transactions*, 41:120–133, 2009.

W.T.I. Miller, R.S. Sutton, and P.J. Werbos. *Neural networks for control*. The M.I.T. Press, Cambridge, MA, 1990.

P.L.M. Van Nyen, J.W.M. Bertrand, H.P.G. Van Ooijen, and N.J. Vandaele. A heuristic to control integrated multi-product multi-machine production-inventory systems with job shop routings and stochastic arrival, set-up and processing times. *Stochastic Modeling of Manufacturing Systems*, pages 253–288, 2006.

K. Papadaki and W.B. Powell. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Research Logistics*, 50(7): 742C769, 2003.

C.D. Paternina-Arboleda and T.K. Das. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simulation Modelling Practice and Theory*, 13:389C406, 2005.

W.B. Powell. The optimizing-simulator: Merging simulation and optimization using approximate dynamic programming. *Proceedings of the 2007 Winter Simulation Conference*, New York:OMNIPress, 2007.

W.B. Powell. Approximate dynamic programming: Lessons from the field. *Proceedings of the 2008 Winter Simulation Conference*, 2008.

W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, Inc., Hoboken, NJ, 2nd edition, 2011.

W.B. Powell and B.V. Roy. Approximate dynamic programming for high dimensional resource allocation problems. *In Handbook of Learning and Approximate Dynamic Programming*, 6:703C710, 2004.

W.B. Powell, A. George, H.P Simio, W. Scott, A. Lamont, and J. Stewart. Smart: A stochastic multiscale model for the analysis of energy resources, technology, and policy. *INFORMS Journal on Computing*, 24(4):665C682, 2012.

J. Qiu and R. Loulou. Multiproduct production/inventory control under random demands. *IEEE Transactions on Automatic Control*, 40(2):350C356, 1995.

A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:211C229, 1959.

H.E. Scarf. *The Optimality of (S, s) Policies in the Dynamic Inventory Problem*. Stanford University Press, Palo Alto, CA, 1959.

H.P. Simio, J. Day, A.P. George, T. Gifford, J. Nienow, and W.B. Powell. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178C197, 2009.

S.R. Smits, M. Wagner, and A.G. de Kok. Determination of an order-up-to policy in the stochastic economic lot scheduling model. *International Journal of Production Economics*, 90:377C389, 2004.

R. Souza, R. Huynh, M. Chandrashekar, and D. Thevenard. A comparison of modelling paradigms for manufacturing line. *IEEE International Conference*, 2: 1253–1258, 1996.

C.R. Sox and J.A. Muckstadt. Optimization-based planning for the stochastic lot scheduling problem. *IIE Transactions*, 29(5):349C357, 1997.

C.R. Sox, P.L. Jackson, A. Bowman, and J.A. Muckstadt. A review of the stochastic lot scheduling problem. *International Journal of Production Economics*, 62:181–200, 1999.

R. Sutton and A. Barto. *Reinforcement learning*. The M.I.T. Press, Cambridge, MA, 1998.

H. Topaloglu and W.B. Powell. A distributed decision-making structure for dynamic resource allocation using nonlinear functional approximations. *Operations Research*, 53(2):281–297, 2005.

H. Topaloglu and W.B. Powell. Incorporating pricing decisions into the stochastic dynamic fleet management problem. *Transportation Science*, 41(3):281C301, 2007.

J.N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16:185C202, 1994.

T.S. Vaughan. The effect of correlated demand on the cyclical scheduling system. *International Journal of Production Research*, 41(9):2091C2106, 2003.

A.F. Veinott. The status of mathematical inventory theory. *Management Science*, 13:475–491, 1967.

M.V Vuuren and E.M.M. Winands. Iterative approximation of k-limited polling systems. *Queueing Systems*, 55(3):161C178, 2007.

M. Wagner and S.R. Smits. A local search algorithm for the optimization of the stochastic economic lot scheduling problem. *International Journal of Production Economics*, 90:391C402, 2004.

D.A. White and D.A. Sofge. *Handbook of intelligent control*. Von Nostrand Reinhold, New York, NY, 1992.

E.M.M. Winands, A.G. de Kok, and C. Timpe. Case study of a batch-production/ inventory system. *Interfaces*, 39(6):552C554, 2009.

E.M.M. Winands, I.J.B.F. Adan, and G.J. van Houtum. The stochastic economic lot scheduling problem: A survey. *European Journal of Operational Research*, 210: 1–9, 2010.

H. Wu, G. Evans, S. Heragu, and K. Rhinehart. Optimization of production and inventory policies for dishwasher wire rack production through simulation. *Proceedings of the 2013 Winter Simulation Conference*, 2013.

P.H. Zipkin. Models for design and control of stochastic multi-item batch production systems, operations research. *Operations Research*, 34(1):91C104, 1986.

CURRICULUM VITAE

NAME: Han Wu

ADDRESS: Department of Industrial Engineering

J.B. Speed School of Engineering

University of Louisville

Louisville, KY 40292

DOB: Chongqing, China - December 07, 1985

EDUCATION: M.S. Operations Research

State University of New York at Buffalo

2009 - 2011

B.E. Industrial Engineering

Southeast University

2004 - 2008

AWARDS: UofL Graduate Student Council Travel Grants

for Conferences (INFORMS, WSC, IIE)

2012 - 2014

The Second Prize of Mechanical Creative Design Competition in Jiangsu province

2006

PROFESSIONAL SOCIETIES:

Institute of Operations Research and the Management Science (INFORMS)

Institute of Industrial Engineer (IIE)

PUBLICATION:

Wu, H., Evans, G., and Bae, K. (2015). "Production control in a complex production system using approximate dynamic programming". International Journal of Production Research. DOI: 10.1080/00207543.2015.1086035

Wu, H., Evans, G., Heragu, S. and Rhinehart, K. (2013). "Optimization of production and inventory policies for dishwasher wire rack production through simulation". Winter simulation Conference, 2666 - 2676

Zhuang J., Saxton G.D. and Wu, H.. (2011). "Publicity vs. impact in nonprofit disclosures and donor preferences: a sequential game with one nonprofit organization and N donors". Annals of Operations Research, ISSN 0254-5330